6001 Normal SATAN probe

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature is generated when an attacking host has run the tool "SATAN" in normal mode against a target host on the network. Other types of attack activity similar to this method may also cause this signature to be generated.

6002 Heavy SATAN probe

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature is generated when an attacking host has run the tool "SATAN" in heavy mode against a target host on the network. Other types of attack activity similar to this method may also cause this signature to be generated.

6050 DNS HINFO Request

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 2

This signature detects an attempt to access HINFO records from a DNS server. This is commonly used by intruders to determine the types of systems on a network.

6051 DNS Zone Transfer

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 1

This signature detects legitimate DNS zone transfers.

6052 DNS Zone Transfer from High Port

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 4

This signature detects a DNS zone transfer with the source port not equal to 53. This is a common network reconnaissance technique used by intruders.

6053 DNS Request for all Records

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 2

This signature detects a DNS request for all records.

6100 RPC Port Registration

SubID Values: RPC program number

Misc Field Info: none

Recommended Alarm Value: 5

This attack attempts to register new RPC services on a target host.

6101 RPC Port Unregistration

SubID Values: RPC program number

Misc Field Info: none

Recommended Alarm Value: 5

This attack attempts to unregister RPC services on a target host.

6102 RPC Dump

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 4

This is an example of network reconnaissance. This is produced by executing a "rpcinfo -p" against a target host.

6103 Proxied RPC Request

SubID Values: RPC program number

Misc Field Info: none

Recommended Alarm Value: 5

This attack exploits a vulnerability of the portmapper by causing it to forward RPC requests to the local system. This may defeat existing authentication mechanisms. The most widespread use of this vulnerability occurs in tricking *mountd* on NFS servers to disclose filehandles.

6150 ypserv Attempt

6151 ypbind Attempt

6152 yppasswdd Attempt

6153 ypupdated Attempt

6154 ypxfrd Attempt

6155 mountd Attempt

6175 rexd Attempt

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

Signatures 6150 through 6175 detect attempts to access the services specified. It is recommended that these services should not be accessed via the Internet.

Regular Expression-Based Signatures

Regular expression- (regex) based signatures are generated from stateful multi-layer datagram contents analysis. In simple terms, this means arranging the contents of packets and doing string matching on the results. Any regular expression up to 64 characters in length is supported, which provides an unlimited number of signatures.

While all regex-based signatures have a **SigID of 8000,** each string must have a unique SubID assigned to it. Numbering SubIDs is left to the individual installing and maintaining NetRanger. Each string entry also specifies the port number and direction of traffic where it should be searched for. For example, in order to detect attackers using the command "help" on mailservers, the regular expression "[Hh][Ee][Ll][Pp]" should be reported only when it is in the datastream going to port 25 on a target host. This drastically lowers the number of false positives.

Performance is the only limitation to the number of strings that can be simultaneously searched for. Analyzing strings directed at the destination port provides substantially better performance than the reverse. For example, malicious activity inside of TELNET sessions can be determined by what an attacker is typing, and not from what appears on the screen. A typical Telnet session has over 95% of the network traffic generated going from the port 23 to the user's screen. In this example, better performance is achieved by only analyzing the remaining 5%.

Following are examples of configurable strings.

NOTE

In the following tables, the numbers in the **Direction** column indicate the following: 1=To Port, 2=From Port, 3=Both. The numbers in the **Occurrences** column indicate how many times the string match occurs before an alarm is sent.

The following string detects users attempting to FTP the password file.

FTP	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	2101	21	1	1	"[Rr][Ee][Tt][Rr][ ]+passwd"

The following strings are used to detect the username and passwords of users in ftp sessions. These also indicate when directories are created/deleted and when files are GET/PUT. These are not security relevant and are provided only to show installers examples of what can be monitored. Capture of user passwords may be a security risk and WheelGroup recommends against it (i.e., what happens if the log file containing cleartext passwords was world readable on a system, printed out, or lost).

FTP	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	2102	21	1	1	"[Mm][Kk][Dd]"
RecordOfStringName	2103	21	1	1	"[Rr][Mm][Dd]"
RecordOfStringName	2104	21	1	1	"[Uu][Ss][Ee][Rr]"
RecordOfStringName	2105	21	1	1	"[Pp][Aa][Ss][Ss]"
RecordOfStringName	2106	21	1	1	"[Rr][Ee][Tt][Rr]"
RecordOfStringName	2107	21	1	1	"[Ss][Tt][Oo][Rr]"

The following strings are used to detect loadmodule/eqvload attacks and exploitation of the Berkeley R-commands. These are representative strings of host-based attacks. Additional strings may be added to detect other host-based malicious activity.

Telnet	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	2301	23	3	1	"IFS[=]+[/]"
RecordOfStringName	2302	23	1	1	"[/]etc[/]shadow"
RecordOfStringName	2303	23	1	1	"[+][ ]+[+]"

The following example shows how to log every HTTP GET on a network. The context info provided with the string will show what URL was used in the GET.

Web	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	8001	80	1	1	"GET[ ]+"

The following example shows how to detect if users are running Microsoft Internet Explorer™. If an organization has a policy of only using Netscape™, detection of MSIE could be enabled generate a security violation.

Web	SIGID	Port	Directions	Occurrences	String Match
RecordOfStringName	8002	80	1	1	"User-Agent.*MSIE"

The following strings are used to detect usernames/passwords of POP3 mail users. As mentioned above, these are not very security relevant.

POP3	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	11001	110	1	1	"[Uu][Ss][Ee][Rr]"
RecordOfStringName	11002	110	1	1	"[Pp][Aa][Ss][Ss]"

The following string is used to detect the name of a user associated with the connection queried in the ident request.

Ident	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	11301	113	2	1	"USER"

The following examples represent another way to detect host based attacks. Use the same strings as in a Telnet session.

rlogin	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	51301	513	3	1	"IFS[=]+[/]"
RecordOfStringName	51302	513	1	1	"[/]etc[/]shadow"
RecordOfStringName	51303	513	1	1	"[+][+]"

NetSentry-Based Signatures

NetSentry-based signatures are generated by reporting packet failures and/or successes from NetSentry filters. When the NSG BorderGuard is used to enforce a specific security policy, an option exists to copy a subset of failed packets from the BorderGuard to the NSX. Each copied packet includes the name of the NetSentry filter that failed the packet. NetRanger matches this name with a pre-defined list of filter names and generates a security violation event record upon a match.

```
filter incom1_ip_spoof_fail
ip_sa mask 0xFF000000 in (10.*.*.*)
copy_to 10.1.5.3 35399
fail;
end
```

The above filter was designed to analyze all packets entering a network. If the source address is from the network, it is an impossible packet and may be part of an IP spoofing attack. The filter will send a copy to the NetRanger/NSX at address 10.1.5.3, and then fail the packet. NetRanger will match the string "Incom1_ip_spoof_fail" and generate a security violation event record. The event record will include the source and destination IP addresses and applicable ports.

Using this NetRanger capability, any filter on the BorderGuard is capable of generating event signatures. Because the BorderGuard allows a large number of highly configurable filters using the NetSentry filter language, the number of potential NetSentry event signatures generated by NetRanger could be considered unlimited. NetSentry filternames for versions 2.0 thru 3.1 of the BorderGuard OS is case sensitive. Version 4 of the BorderGuard OS and the ERS report filternames in uppercase. If the original filtername is "finger_fail", then NetRanger must be configured to look for the filtername "FILTER_FAIL".

All NetSentry-based event records have a SigID of 10000. Each defined filtername in NetRanger must have a unique SubID assigned to it. Numbering SubIDs is left to the individual installing and maintaining NetRanger. The example security filters that come with NetRanger implement the following:

```
incom1_ip_spoof_failIP spoofing attacks
incom1_ip_source_route_failIP source routing attacks
incom1_tcp_frag_header_failFragmented TCP header attacks
incom1_tcp_small_frag_failUndersized TCP header attacks
incom1_tcp_failGeneric failed TCP connections
incom1_udp_failGeneric failed UDP packets
incom1_ftp_failFailed FTP attempt
incom1_telnet_failFailed TELNET attempt
incom1_smtp_failFailed e-mail attempt
incom1_dns_failFailed Domain Name Server attempt
incom1_gopher_failFailed gopher attempt
incom1_finger_failFailed finger attempt
incom1_www_failFailed WWW attempt
incom1_pop2_failFailed POP2 mail attempt
incom1_pop3_failFailed POP3 mail attempt
incom1_rpc_failFailed RPC attempt
incom1_auth_failFailed identd attempt
incom1_nntp_failFailed network news attempt
incom1_ntp_failFailed network time attempt
incom1_exec_failFailed rexec attempt
incom1_login_failFailed rlogin attempt
incom1_cmd_failFailed rsh attempt
incom1_printer_failFailed printer attempt
incom1_ntalk_failFailed network talk attempt
incom1_uucp_failFailed UUCP attempt
incom1_x11_failFailed X11 attempt
incom1_udp_dns_failFailed DNS packet
incom1_tftp_failFailed TFTP packet
incom1_udp_rpc_failFailed RPC packet
incom1_snmp_failFailed snmp packet
incom1_syslog_failFailed syslog packet
```

5 THE SECURITY ANALYSIS PACKAGE

Collection, Management, and Analysis of NetRanger Data

This chapter presents the following information on the NetRanger Security Analysis Package (SAP):

- *High Level Overview*
- *Functional Overview*
- *Installation*
- *SAP Tutorial*
- *SAP Reference*
- *File Management Tokens*

High Level Overview

The NetRanger Security Analysis Package (SAP) provides data collection, management, and analysis services on both NSX and Director systems. These services represent a key operational component of NetRanger. Figure 5.1 illustrates the relationship among the parts.

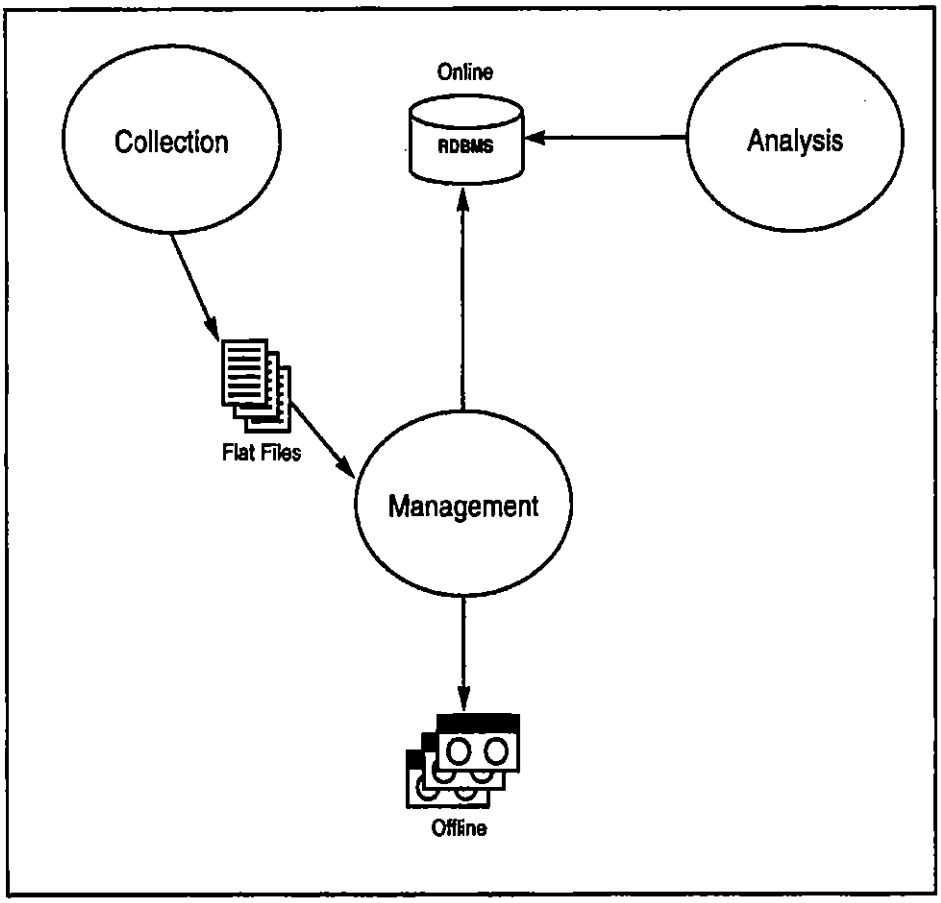

Figure 5.1: Overview of SAP Functions

Collection

SAP collects two basic forms of data: high-level event data and detailed binary IP session data. SAP uses the two types of log files to capture these types of data:

- **Event logs,** which capture *NSX alarms, system commands,* and *system errors.* These are ASCII files written to the /usr/nr/var directory with the naming convention of log.<YYYYMMDDHHMM>. By default, level 2-5 events are written to event logs on a Director system, and low-level 1 events are written to an event log on the NSX system.

- **IP Session logs,** which capture all of the incoming and outgoing TCP packets associated with a specific connection. By definition, these logs contain binary data. They are written to an NSX Sensor's /usr/nr/var/iplog directory with the naming convention of iplog.<src IP address>.

By default, level 1 Event and IP Session logs are left on an NSX Sensor until they are required. This prevents the relatively large amounts of data associated with these types of events from impeding NetRanger communications during periods of network load.

Even though the SAP is designed to transfer NetRanger data into industrial strength databases, both types of data are initially written to flat files for two reasons: ***speed*** and ***fault tolerance.*** Data can be written to a flat file much faster than to a database, and access to flat files is guaranteed as long as the host system is operational. Databases, on the other hand, are subject to unpredictable load, and contain too many access layers to be truly fault tolerant—if the database is down, you've got nowhere to go.

Management

The three main goals of SAP data management are to automatically

- collect data in a fast, fault-tolerant manner;
- stage that data onto database management and archival systems; and
- prevent disk systems from filling up.

These services are easily configured and monitored via **nrConfigure.**

Analysis

Once data is loaded into a database, it can be analyzed for patterns and trends. Status reports relating to network activity and vulnerabilities can also be generated. Although any number of different third-party tools can generate these types of output, SAP is shipped with a small but comprehensive collection of Oracle SQL*Plus queries that show how event data can be analyzed relative to different perspectives, such as time versus events. These queries are described in *SAP Tutorial.*

Note that although Collection and Management are provided by default on an NSX Sensor, only Collection services are provided by default on Director systems. SAP is bundled in this manner for two reasons:

1. The integrity of the file system on an NSX Sensor is a core operational requirement, and the SAP Collection and Management services fulfill this requirement.

2. The Management and Analysis services on a Director system are designed to work with large volumes of data stored in a Relational Database Management System (RDBMS). These services are bundled as a separate component to help keep costs down for those users who do not require this level of functionality.

Table 5.1: Availability of basic SAP services

	Collection	Management	Analysis
NSX	standard	standard: file only	not applicable
Director	standard	optional: file & data	optional

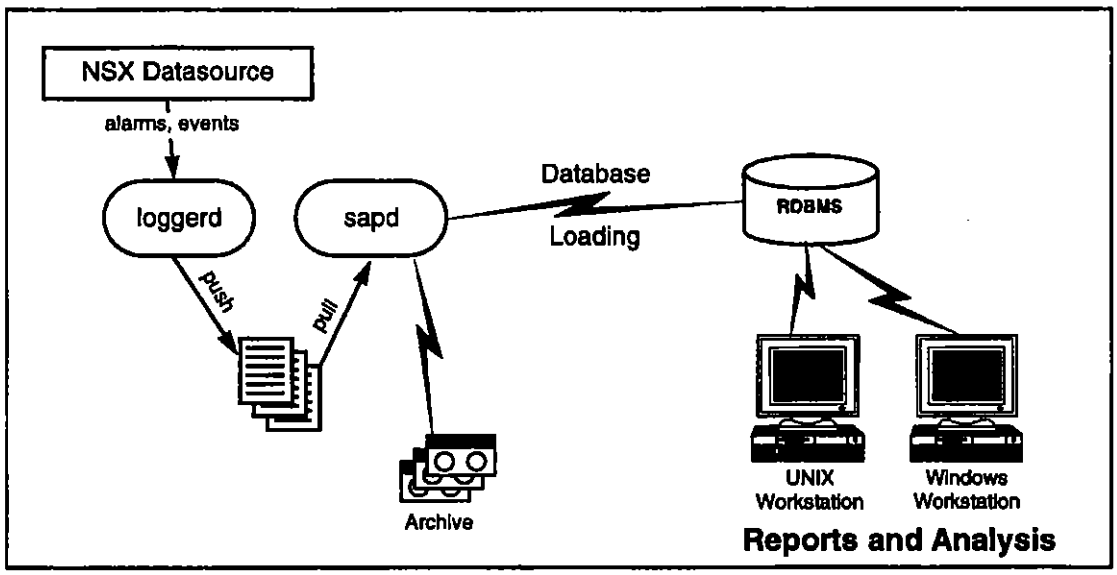

Figure 5.2: Data Collection, Management, and Analysis

Functional Overview

NetRanger uses a simple **push-pull** process to migrate data from flat files to a database or data archive. Figure 5.3 diagrams the push-pull process for both types of log files.

The *push* process uses the *loggerd* service to write *event*, *command*, and *error* notifications into a single flat file in /usr/nr/var. This file is *serialized* based on configurable size and time thresholds set via **nrConfigure**. Once the current log file exceeds one of these thresholds, it is replaced by a fresh log file and moved to /usr/nr/var/new.

The *pull* process is driven by *sapd* service, which relies on its own size and time thresholds. Note that multiple log files accumulate in /usr/nr/var/new whenever *loggerd* serializes log files at a faster rate then *sapd* writes to a database. Fortunately, high data rates tend to be episodic, and the SAP is designed to recover from such loads by processing all of the files in /usr/nr/var/new until the directory is empty. *sapd* pulls the oldest file from /usr/nr/var/new into /usr/nr/var/tmp and executes database load procedures specified by the user. After the log file is successfully loaded into a database, it is placed in /usr/nr/var/dump where it is processed by a user-defined purge process.

This basic push-pull process applies to IP session logs as well as event logs. The primary differences are that IP session logs are written to /usr/nr/var/iplog rather than /usr/nr/var/new, and they are automatically placed in /usr/nr/var/dump after a configurable amount of time.

A number of different data management scenarios are presented in the *SAP Tutorial* section.

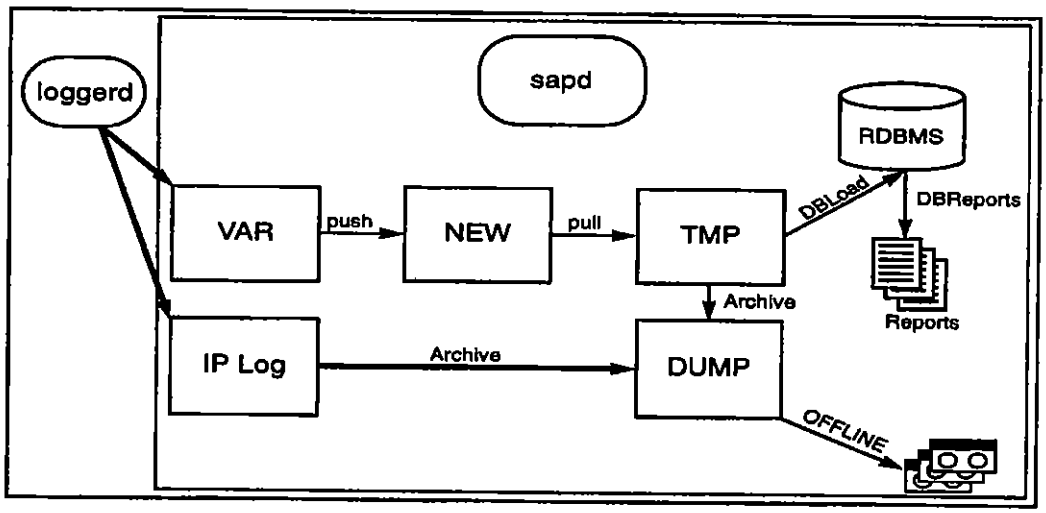

Figure 5.3: The Push-Pull Process

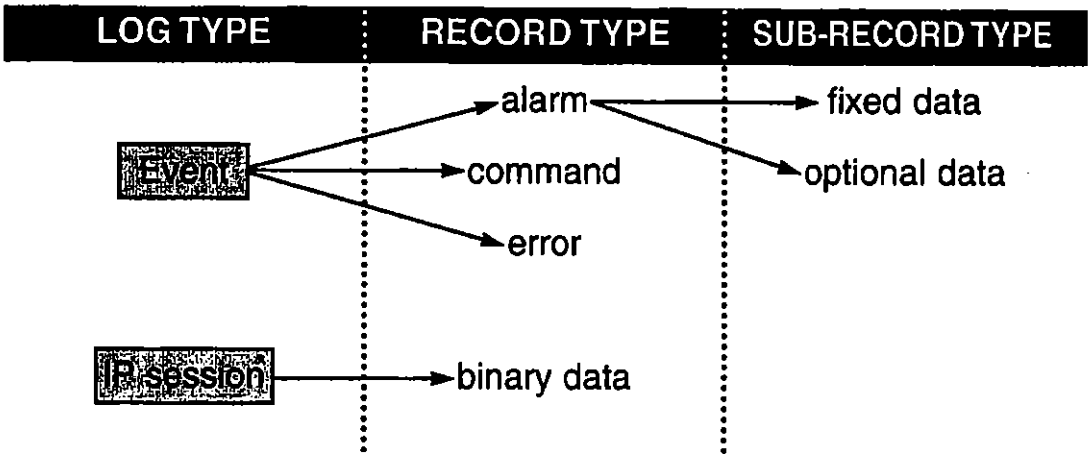

Figure 5.4: Basic Log File Data Relationships

Collection

Event Alarm Data

As noted in the introduction, SAP generates two basic types of log data: Event and IP Session data. Event alarm data is unique in that its record format contains *optional* as well as a *fixed* components.

Fixed Alarm Data

Fixed alarm data includes the following information:

- Which sensor detected the event.
- When the alarm was generated.
- The type of alarm.
- The source and target of the event.

The exact record format for the fixed portion of alarm data is defined in Table 5.5 of the *SAP Reference* at the end of this chapter.

Optional Alarm Data

The *optional* data field contains *ad hoc* information that cannot be described by the fixed portion of the alarm record format. Optional data is *ad hoc* because it depends on the type of alarm record it is associated with. Many context-based alarms, such as port sweeps and ICMP requests, currently do not use this part of the alarm record. Others, such as Security Violation and DNS Request alarms, use this portion of the alarm record to record such information as the name of the policy filter or the request string associated with the event.

A third type of optional data is *Context* data. Not to be confused with context-based alarms, this optional data contains a snapshot of the incoming and outgoing network traffic that leads up to the generation of any type of alarm that includes the exchange of textual data. This data can then be used to reconstruct the events that led up to the unauthorized event.

This type of data cross-cuts the context-content alarm signature classifications defined in Chapter 4. It is generated for such context-based alarms as Telnet and HTTP-Server connection requests as well as such content-based signatures as Sendmail and User-Defined signatures with a SIGID of 8000.

A maximum of 256 bytes of incoming and 256 bytes of outgoing network traffic is associated with an alarm record. The Director's **Security->Show->Context Data** menu option allows you to view the context data for certain alarm levels. The data can also be automatically loaded into the ***nr_log_context*** table at the same time the fixed data is loaded into ***nr_log_alarm.***

As one might expect, there is a performance penalty associated with the logging of context data. SAP provides two configuration parameters to help control this overhead:

MinContextLevel is a *loggerd* token that allows you to specify the minimal alarm level for logging optional context data. For example, context data associated with level 1-3 alarms will not be written to a Director's Event log file when this token is set to "4." Note that this token does not affect access to this information from the Director console. Users can still view this information for alarms that fall below the MinContextLevel threshold.

SAP_EXCLUDE_CONTEXT. The *sapd* process relies on a script named **load_run.sh** to determine exactly what levels and types of event data should be exported onto a database. Context data is not loaded into the ***nr_log_context*** table when this variable is set.

Log File Serialization

As noted earlier, serialization of log files is based on time, size, or combined time/size thresholds specified via **nrConfigure.** The *time* threshold is specified via *loggerd's* **NumberOfSwitchMinutes** token; *size* is specified by the **NumberOfSwitchBytes** token. Appropriate threshold values depend on such factors as network activity, and the amount of disk space allocated for log files.

In the following example, all but one file was serialized on the basis of a time threshold of **one hour**. *log.199611070819* was serialized ahead of schedule because it exceeded the 100-kilobyte *size* threshold before the next hour had passed.

Filesize	Date	Time	Filename
61058	Nov 7	06:19	log.199611070519
60524	Nov 7	07:19	log.199611070619
70008	Nov 7	08:19	log.199611070719
103730	Nov 7	08:40	log.199611070819
76926	Nov 7	09:40	log.199611070840
83476	Nov 7	10:40	log.199611070940

Fail-Safe Features

One of the primary requirements of *loggerd* is that it perform in a fail-safe manner under heavy load. These fail-safe measures currently include safeguards against runaway serialization and loss of files due to overwriting. Users are also able to perform manual serialization.

Runaway Serialization

In order to prevent *loggerd* from creating a large number of log files under heavy load, and possibly exceeding the file management limits of the operating system, *loggerd* makes sure that the current log file remains open for at least one minute. In the following example, log files of 259K and 177K were created between 10:14 and 10:16 because the one-minute threshold had to be met in spite of the size threshold.

Filesize	Date	Time	Filename
100358	Nov 11	09:49	log.199611110916
101432	Nov 11	10:14	log.199611110949
259555	Nov 11	10:15	log.199611111014
177152	Nov 11	10:16	log.199611111015
100307	Nov 11	10:44	log.199611111016

Loss of Files from Overwriting

In addition to preventing runaway serialization, *loggerd* is designed to prevent files from being overwritten when NetRanger services are *started and stopped more than once a minute.* Although uncommon, this type of activity can occur during system maintenance. In this situation *loggerd* appends a sequential identifier to each log file name.

In the following example, the NetRanger daemons were started and stopped four times within a single minute (16:44). As a result, *loggerd* appended the numbers 1, 2, and 3 to the end of the last three files.

```
Filesize    Date     Time     Filename
  178       Nov 8    16:44    log.199611081644
  313       Nov 8    16:44    log.199611081644.1 <--
    0       Nov 8    16:44    log.199611081644.2 <--
  306       Nov 8    16:44    log.199611081644.3 <--
```

Manual Serialization

The current event log file can also be serialized on demand by issuing an **nrexec** command against *loggerd's* **SwitchFile** token. Size and time thresholds are reset at that point.

Management

sapd

sapd is the NetRanger daemon that launches database load or archive processes based on user-defined **conditions** and **actions.** Essentially, an action is launched when its associated condition is satisfied. Condition-action pairs are known as ***triggers.***

Multiple ***triggers*** can be defined via **nrConfigure,** which are evaluated by *sapd* in the order in which they are defined. This supports prioritized processing of triggers. For example, the most important criterion for processing log files might be an upper limit on the amount of disk space that log files are allowed to consume. The next criterion might be a fixed time interval.

Conditions

Much like *loggerd*, *sapd* conditions are based on filesystem and time thresholds.

Filesystem conditions can be based on any of the following thresholds:

- **Absolute number** of log files in the source directory.
- **Cumulative size** of all of the log files in the source directory.
- **Overall percentage of the disk file partition consumed.**

Time conditions are based on either of the following:

- An **elapsed time** interval.
- A **specific day or time** of day.

Actions

The actions that can be launched by *sapd* include

- **DBLOAD**, which loads the contents of an ASCII file into a database management system, such as Oracle or Remedy ARS.
- **ARCHIVE**, which loads a log file into a user-defined destination, such as tape storage. This action includes a built-in *purge* facility that removes the oldest file in /usr/nr/var/dump when a user-defined disk utilization threshold is reached. A bulk-purge facility can also be configured to move a set of log files instead of one at a time. Note that *sapd* will not evaluate another condition until it has completed the current ARCHIVE action.
- **BATCH**, which is a version of ARCHIVE that operates in the background and allows *sapd* to continue evaluating other conditions without *blocking*. This action is intended primarily for launching database reports, which can execute in parallel with other actions. ARCHIVE is better suited for controlling archiving actions, which generally cannot proceed in parallel unless the NSX or Director has access to multiple archival devices.
- **NOTIFY**, which repeatedly launches an action based on a *NotifyInterval* so long as the launch condition holds true. Although of general-purpose construction, this action type was designed to generate e-mail messages when the NSX or Director file system exceeds user-defined thresholds.

sapx

sapx is the default database load program shipped with the SAP package. It currently supports Oracle and the Remedy ARS system. Event log data can be exported to other database management systems, such as Informix by using their native bulk data load utilities. The rules files for Oracle's SQL*Loader utility are included in /usr/nr/bin/sapd/skel for reference.

sapx requires the following parameters in order to load data into a database:

- **LogFileName,** which identifies the directory location and name of the log file.
- **SourceType,** which identifies the event record types to pass onto the target database. Event records are uniquely identified by a numeric marker that is either 2 (error), 3 (system command), or 4 (alarm). *sapx* also supports the record type "5". This is a reserved record identifier for alarm signatures, which are only loaded during initialization or manual update of a database. Also note that although *sapx* can load NetRanger error, command, alarm, and signature records into Oracle, it only loads alarm records into Remedy ARS systems.
- **TargetType,** which identifies the type of database to load the log file in to. The value for Oracle is "1". The value for Remedy ARS is "3".

The *SAP Reference* section provides more information on *sapx*.

Analysis

sapr

As noted in the *High Level Overview* section, SAP is shipped with a small collection of SQL queries that generate simple columnar reports. These reports are referred to as the SAP Reports, or *sapr,* and demonstrate how NetRanger intrusion detection data can be viewed from three basic perspectives: **Space, Time,** and **Events.**

Each of the *sapr* perspectives is effectively a *data dimension* that contains subordinate elements, or *sub-dimensions.* For example, *Time* can be broken down into years, months, days, and so on. *Space* starts with IP addresses and ends with specific ports on a host. Imposing these type of hierarchical structures on NetRanger's event data makes it easier to move from high to low levels of detail as well as shift from one perspective to another. Changing the level of detail is commonly known as **data drilling,** while changing perspective is referred to as **data pivoting.**

Both the dimensions and the associated *sapr* queries are presented in the *SAP Reference* section.

Installation

This section summarizes the steps required to deploy an operational SAP on both the **NSX** and **Director**.

NOTE

HP-UX systems require one change in NetRanger's .profile. Please refer to Appendix D: DBMS REQUIREMENTS AND SETUP for information on changing the stty setting for the @ character.

Before You Install the Security Analysis Package

Collect the following information before you install and configure this version of the SAP:

- Logging Profile
- Serialization Thresholds
- File Management Thresholds
- Target Archive and Dump Destinations
- Notification Intervals, Conditions, and Destinations

Logging Profile

As explained at the beginning of this chapter, there are two basic types of data that can be logged by an NSX sensor: *Events* and *Binary IP Session data*. Although enabling IP Session logs is a simple yes/no proposition, Event logs require a data resolution rule. For example, a possible logging profile may include one of the following:

- Alarms must be level 2 and above
- Information must be level 1 and above
- Full Information must be level 1 with *sensord's* **LevelOfLogging** token set to 3

Serialization Thresholds

As noted previously, *loggerd* serialization thresholds can be time-, size-, or time/size-based. The minimum *time* threshold is one minute. The minimum log *size* threshold is one kilobyte. Some possible settings include:

- **Every minute.** This setting should be used when you want event log data processed as fast as possible, or to store event data in smaller files.
- **Every hour.** This assures that *loggerd* serializes the current log file on an hourly schedule.
- **Every 10 minutes** or **when the log file exceeds one megabyte in size**.

File Management Thresholds

In order to prevent an NSX Sensor's file system from locking up, file management thresholds must be defined for the following SAP holding areas under /usr/nr/var: **new, iplog,** and **dump.** As with *loggerd,* these thresholds can be specified based on time, size, or a combination of both. The options consist of the following:

- **DirFiles**, which designates a **maximum number of files** in the holding directory (e.g., maximum of 5 files).
- **DirSize**, which designates a **maximum number of kilobytes** consumed in the holding directory (e.g., 2000 KB = 2 MB).
- **DirUsed**, which designates a **maximum percentage of the disk partition** consumed by the holding directory (e.g., 30%). Another way of thinking about this threshold is percentage of the disk partition that is *unused.*
- **CronTime**, which designates a **time of Day** (e.g., every day at 1AM).
- **IdleTime**, which designates an **elapsed Time Interval** (e.g., every 30 minutes).

Each NSX Sensor is shipped with default thresholds, which can be found in /usr/nr/etc/sapd.conf.

Target Archive and Dump Destinations

Although data can be archived and dumped to a storage device (such as a DAT tape), the *default archive process* compresses log files and then moves them to /usr/nr/var/dump. The *default dump process* deletes the oldest file.

- **Archive:** directory or device
- **Dump:** directory, device, or purge process

Notification Interval, Conditions, and Destination

As stated earlier, the SAP can generate e-mail notifications on a regular interval so long as a condition, such as violation of a file system threshold, is true. For example, you may wish to be notified when your NSX /usr/nr/var filesystem exceeds 75% of the disk drive's capacity, or when there is more than 250 megabytes of data in the /usr/nr/var/dump directory. The following information must be defined in order for this facility to work properly:

- **E-mail or pager/e-mail address** (admin@command.com).
- **Conditions to initiate the notification** (see the serialization thresholds listed above).
- **The notification interval,** which is how often to send a notification while the condition is true (e.g., every 20 minutes).

Installing and Configuring SAP on an NSX

The nrConfig utility guides you through a series of menus that assist with installation and configuration of the SAP package.

The first menu in nrConfig is the Feature Selection Menu, and looks something like the following:

```
FEATURE SELECTION MENU
Choose what features you want ENABLED on this host.
(Choosing an 'ENABLED' feature will disable it.)
        1 - NSX
        2 - Director
        3 - Logging
        4 - Database Reporting
        5 - File Management
        6 - Event Paging
        7 - Postoffice Routing
        8 - Configuration Control
    Enter - CONTINUE
Feature # >
```

From this menu, you can enable and disable various features. When you select a feature from the menu, it is prefixed with ENABLED to show that it has been selected. To disable a selected feature, simply choose it from the menu again.

To install and configure SAP on an NSX sensor, ensure that item 2 in the Feature Selection Menu (Director) is **not** enabled. Enable logging, database reporting, and file management features. The menu will look something like the following:

```
FEATURE SELECTION MENU
Choose what features you want ENABLED on this host.
(Choosing an 'ENABLED' feature will disable it.)
ENABLED 1 - NSX
        2 - Director
ENABLED 3 - Logging
        4 - Database Reporting
ENABLED 5 - File Management
        6 - Event Paging
ENABLED 7 - Postoffice Routing
        8 - Configuration Control
    Enter - CONTINUE
Feature # >
```

Next, press Enter to return to the Main Menu, which will look like the following:

```
MAIN MENU                    NetRanger Configuration Version 1.3.1
Choose what Section you want to configure.
         1 - Select Features
         2 - Host Address Information
         3 - Sensor Configuration
         4 - N/A (Database Configuration)
         5 - Source Configuration
         6 - Destination Configuration
         7 - Postoffice Router Configuration
         8 - N/A (Sleeve Configuration)
         9 - Clear Temporary Configuration Files
        10 - Generate Temporary Configuration Files
        11 - Edit/Review Temporary Configuration Files
        12 - Review Temporary Configuration Files
        13 - Commit Temporary Configuration Files
     Enter - EXIT
     Section # >
```

Press 4 for Database Configuration, which will guide you through the process of entering local *sapd* token values. The menu for Database Configuration looks like the following:

```
 Database USER ID :
Database PASSWORD :
    Notify Person :
DATABASE CONFIGURATION MENU
        1 - Database USER ID
        2 - Database PASSWORD
        3 - Notify Person
    Enter - CONTINUE
Select Database Configuration Field # >
```

Press 1 to enter a Database USER ID, and so on. Sample entries are illustrated in the following table:

Database USER ID	netranger@DBserver
Database PASSWORD	thePassword
Notify Person	john.doe@xyzcorp.com

After entering the local *sapd* tokens, press Enter to return to the Main Menu. From this menu, finalize the configuration by selecting the following:

```
10 - Generate Temporary Configuration Files
13 - Commit Temporary Configuration Files
```

At this point, please confirm the following information:

That this directory/file...	Contains the following...
/usr/nr/etc/daemons	nr.loggerd and nr.sapd
/usr/nr/etc/destination **or** /usr/nr/etc/smid.conf	nr.loggerd
/usr/nr/etc/loggerd.conf	defaults only
/usr/nr/etc/sapd.conf	defaults only

The nrConfig portion of the SAP installation for the NSX is now complete. The next step in the process involves using a command-line editor (like vi) to configure the default settings in both /usr/nr/etc/loggerd.conf and /usr/nr/etc/sapd.conf.

Configuring loggerd.conf involves setting the tokens that control serialization of log files by size (**NumberOfSwitchBytes**) and time (**NumberOfSwitchMinutes**), and setting **MinContextLevel**, a token that *loggerd* uses to filter context data out of event records.

After configuration of loggerd.conf, it is a good idea to generate some network activity that *sensord* will send as events. This will confirm that *loggerd* is properly configured to write Event and IP logs.

A key component of *sapd* is the management of the file system. sapd.conf contains the following default file management processes.

Process	sapd.conf File Management Triggers
Event Archiving	`FM_Action Archive_New /usr/nr/bin/sap/dump.sh $DirPath NEWLOG log.*`
IPLOG Archiving	`FM_Action Archive_Iplog /usr/nr/bin/sap/ipdump.sh $DirPath IPLOG`
Error Traps	`FM_Action Trap_Var /usr/nr/bin/sap/dump.sh $DirPath NRVAR errors.* messages.*`
Purging Dump	`FM_DirUsed DUMP_Purge 90 /usr/nr/var/dump Dump_Purge`
System Notifications	`FM_Action Ntfy_Urgent /usr/nr/bin/sap/notify.sh $DirPath $DirUsed $Notify1 URGENT`
	`FM_Action Ntfy_Purge /usr/nr/bin/sap/notify.sh $DirPath $DirUsed $Notify1 Purging`
	`FM_Action Ntfy_Warn /usr/nr/bin/sap/notify.sh $DirPath $DirUsed $Notify1 WARNING`

You may want to fine-tune the threshold parameters to better accommodate your disk space and system requirements.

Installing and Configuring SAP on a Director

Installation and configuration of SAP on a Director follows the same process as installation and configuration of SAP on an NSX sensor. Ensure that you ENABLE the Director instead of the NSX on nrConfig's Feature Selection Menu, so that it looks like the following:

```
FEATURE SELECTION MENU
Choose what features you want ENABLED on this host.
(Choosing an 'ENABLED' feature will disable it.)
        1 - NSX
ENABLED 2 - Director
ENABLED 3 - Logging
ENABLED 4 - Database Reporting
ENABLED 5 - File Management
        6 - Event Paging
ENABLED 7 - Postoffice Routing
        8 - Configuration Control
    Enter - CONTINUE
Feature # >
```

Follow the rest of the steps outlined in *Installing and Configuring SAP on an NSX* until you have finished configuration of loggerd.conf and sapd.conf.

When configuration of these files is complete, you are ready to set up the Director's DBMS functionality. You have the following data management options:

- Oracle DBMS
- non-Oracle DBMS

Oracle DBMS Setup

Oracle is the default SAP database. Setting up the Oracle database consists of the following steps:

1. **Database server setup.**
2. **Database schema setup.**
3. **Database load setup.**
4. **Database reports setup and customization.**

Database Server Setup

Please refer to Appendix D: DBMS REQUIREMENTS AND SETUP for more information on setting up the database server.

Database Schema Setup

Use the utility nrdb_master_create in /usr/nr/bin/sap/sql/skel/ to initialize your database schemas. You will be prompted for the following information:

- database user
- password
- location

Enter your database user name and password. For location, you have a choice of either local or remote. If you have a local DBMS, enter nothing at the `Remote DB Service` prompt. If you have a remote DBMS, enter the service name preceeded by an @ character.

Please refer back to the *Notification Interval, Conditions, and Destination* section on page 15 for examples of values to enter.

Database Load Setup

Use the SAP defaults unless you want to exclude elements. To do this, edit the load_run.sh file in /usr/nr/bin/sap and comment out those elements you wish to **include** in the reports. In the following example, the SAP_EXCLUDE_ALARM element has been commented out, and will therefore be included in generated reports. The other elements will be excluded from the reports.

```
#export SAP_EXCLUDE_ALARM=
export SAP_EXCLUDE_TCPCONN=
export SAP_EXCLUDE_CONTEXT=
export SAP_EXCLUDE_ALARM_1=
```

When this portion of the Oracle DBMS setup is complete, it is a good idea to verify that *sapd* will autoload staging event logs. You can do this by using instrumentation (described in *How to Use Instrumentation* in the *SAP Tutorial*) to observe the autoloading after log data has been loaded by *sapd*, and when you run queries.

Database Reports Setup and Customization

The queries shipped with the SAP generate basic reports, which are accessed via three different .sql entry points:

- event.sql
- space.sql
- time.sql

In turn, each of these queries gives access to a number of different subordinate queries, such as event1.sql, event2.sql, and so on. The query you run depends on the answers you give to the initial (event/space/time) query. Basically, the greater the number of the subordinate query, the more detail is returned.

You can customize these queries to better meet your reporting needs by editing `event.sql`, `event1.sql`, and `event2.sql` in /usr/nr/bin/sap/sql. Below is the interactive component of an `event` query. User input is bolded.

```
SQL> @event
EVENT DIMENSION QUERIES
=======================================
1    alarm level summary
2    alarm signature summary
3    alarm signature with string data summary
These queries summarize NSX alarms with a primary focus
on the EVENT dimension.
You can filter the data returned to you with the following
criterion:
            ORGANIZATION_NAME
            MIN_EVENT_DATE
Please enter your desired value at the following prompts:
select query (1,2,3)>1
organization name (%)>%
minimum event date (MM/DD)>5/1
```

The results of the above query follow:

```
event1: alarm level summary
organization_name = %
minimum_event_date = 1997/5/1
Org Name         Level       From  To    Count       Recent
---------------  ----------  ----  ---   ----------  -----------
Auto Mart         3          OUT   IN    33          05:01 15:08
Data Pros         5          IN    OUT   3           05:02 08:45
Net King          5          IN    IN    15          05:02 11:21
Net King          3          OUT   OUT   4           05:01 19:15
Net King          3          IN    IN    2           05:01 15:03
Net King          3          IN    OUT   79          05:02 09:28
WheelGroup        3          IN    IN    231         05:02 09:37
WheelGroup        3          IN    OUT   176         05:02 09:37
WheelGroup        2          OUT   IN    9110        05:02 12:15
WheelGroup        2          IN    IN    18          05:01 14:06
```

In the example, the alarm **level** activity from four **organizations** is profiled. **From** refers to the source of the alarm direction (in relation to the trusted internal network) and **To** indicates the destination of the alarm direction. The **Count** is the number of alarms that match the specific criteria for each line in the generated report. **Recent** gives a timestamp of the last alarm activity for that criteria.

To change the scheduling of reports, customize the SAP triggers. These triggers can be found in /usr/nr/etc/sapd.conf.

SAP reports are written to /usr/nr/var/db. Each file contains a header with selection criteria and datestamp. You can customize the reporting process in several ways:

- The files DAY, WEEK, and MONTH in /usr/nr/bin/sap/sql are templates for report batches. They can be altered with various criteria to generate different sets of reports.
- New templates can be added to supplement the daily, weekly, and monthly templates. This will require a simple trigger modification or addition in sapd.conf.
- The actual SQL DML files (event1.sql, event2.sql, etc.) in /usr/nr/bin/sap/sql can be changed to alter the selection, sorting, summarization, and display of data.

Non-Oracle DBMS Setup

If you are setting up a generic SQL-based RDBMS, follow these steps:

1. **Database server setup.** Please refer to Appendix D: DBMS REQUIREMENTS AND SETUP for more information on setting up the database server.
2. **Database schema setup.** Follow the same procedure outlined in the *Oracle Database Schema Setup*, except the files for setup reside in /usr/nr/bin/sap/sql/skel.
3. **Database load setup.** Setting up the database loader involves choosing a loader (*sapx*, Oracle's own loader, or another load utility), and customizing the triggers in /usr/nr/etc/sapd.conf that schedule loading.
4. **Database reports setup and customization.** Follow the same procedure as with *Oracle Database Reports Setup and Customization*.
5. **Scheduler setup.** Ensure that the DBLoad and BatchReports tokens are properly set (in /usr/nr/etc/sapd.conf) to meet your needs.

If you are setting up a Remedy ARS trouble ticketing system, follow these steps:

1. **Copy the** `load_run.sh.remedy` **file from the /usr/nr/bin/sap/skel directory into /usr/nr/bin/sap, and rename it** `load_run.sh` **from /usr/nr/bin/sap.**
2. **Ensure that the DBLoad token is properly set in /usr/nr/etc/sapd.conf.**
3. **For logon requirements, edit the following tokens in /usr/nr/etc/sapd.conf: DBUser2, DBPass2, DBAux1, DBAux2, and DBAux3.**
   - DBUser2—Enter the Remedy user name (must not contain spaces).
   - DBPass2—Enter the Remedy user password (cannot be blank).
   - DBAux1—Enter "C" for this value (refers to Remedy's LANG variable).
   - DBAux2—Enter the name of the server running Remedy.
   - DBAux3—Enter NR_Alarm_Schema.
4. **To create a schema, use the Remedy GUI to import the** nr_alarm_schema.def **file from the /usr/nr/bin/sap/skel directory.**

SAP Tutorial

The goal of this section is to introduce you to **instrumentation**, **triggers**, and **polling intervals**. The section after this, *SAP Reference*, will build on the knowledge gained in this tutorial.

How to Use Instrumentation

The first thing you'll want to learn is how to use the SAP instrumentation to perform system monitoring of the following:

- Management
- File System
- Actions

You can use instrumentation to view the following: a summary of the FileMgmt token settings, a summary of the /nr/var directories, or the current configuration of actions.

Viewing FileMgmt Instrumentation

There are two ways to view FileMgmt status: Brief mode and Full mode. Brief mode provides a summarized view of the FileMgmt information, and includes overall status and uptime, a summation of errors, and a trigger history. Full mode is a superset of Brief mode, and also includes Scheduler status, normal summation, and trigger configuration.

To set for Brief mode, type the following command:

```
nrset 10007 . . 1 DisplayMode Brief
```

For Full mode, simply replace `Brief` with `Full` in the above command line.

To see what Brief mode instrumentation looks like, set your mode to Brief, and type in

```
nrget 10007 . . 1 FileMgmt
```

Something like the following information is returned:

```
Overall Status = ERROR Archive1 ;
SAPD up: 1 minute 2 seconds
current time: 05/12 11:10
_____________________________________________________________
ERR total:1  last: 05/12 11:10  Archive1
_____________________________________________________________
Trigger History
Status  OK    Error  Trigger Name   Action       Last Run
------- ----- -----  -------------- ------------ -----------
Normal  1     0      OracleLoad      DBLoad_Proc   05/12 11:09
ERROR   0     1      Archive1        Archive_Proc  05/12 11:10
Normal  0     0      Batch_Reports   Batch_Proc
```

Following is an example of the information that would be returned if the mode were set to Full. Those sections that are in Full mode and not Brief mode have been bolded.

```
Overall Status = ERROR Archive1 ;
SAPD up: 4 minutes 0 second
current time: 05/12 11:13
PollingDelay = 1   DisplayMode = Full
Master Processing: IDLE
Batch Processing
(Batch_Reports) (pid 3246) (running 10 seconds)

ERR total:4  last: 05/12 11:13  Archive1

NORMAL  total:6  last: 05/12 11:09  OracleLoad

Trigger History
Status  OK    Error  Trigger Name    Action       Last Run
------- ----- -----  --------------- -----------  -----------
Normal  1     0      OracleLoad      DBLoad_Proc   05/12 11:09
ERROR   0     4      Archive1        Archive_Proc  05/12 11:13
Normal  0     0      Batch_Reports   Batch_Proc    05/12 11:11

Trigger Configuration
Trigger Name    Action       Condition
--------------- ------------ --------------------------------
OracleLoad      DBLoad_Proc  DirFiles = 1 in /usr/nr/var/new
Archive1        Archive_Proc IdleTime(60) next: 05/12 12:13
Batch_Reports   Batch_Proc   CronTime    next: 05/14 21:00
```

Both Brief and Full modes contain an `Overall Status` line (the 1st line in the above example). This line indicates that *sapd* keeps track of its overall status; that is, if a launched action returns indicating an error condition, *sapd* knows that an error has occurred. In fact, the above example shows that an error has occurred:

```
Overall Status = ERROR Archive1 ;
```

If an error happens, *sapd's* status will remain in an ERROR state until one of the following occurs:

1. **The action is launched again and it returns a Normal (non-error) indicator.**
2. **The user executes the token UserStatus with the value "OK."**

For example, let's say that right before a scheduled staging of files to a database, the server goes down. If the server is down, then the staging does not occur at the scheduled time, and this would cause an Error to appear in the Overall Status line.

However, if the server is brought back on line before the next scheduled staging of data, and the staging occurs, then the Error would be replaced by a Recover in the Overall Status line, like so:

```
Overall Status = Recover Archive1 ;
```

Once this problem has been investigated and rectified, you can simply reset all of the status instances by typing the following command:

```
nrset 10007 . . 1 UserStatus OK
```

Once reset, the Overall Status line will look like the following:

```
Overall Status = Normal
```

Using VarSize

The VarSize token allows you to quickly assess the state of the SAP file system on an NSX or Director. Unlike FileMgmt, VarSize has only one mode of operation. To execute it, use the following command line:

```
nrget 10007 . . 1 VarSize
```

This will display a summary of all the /nr/var directories, which will look something like the following:

```
filepath          files   bytes    oldest        newest
/usr/nr/var       4       168782   05/07 16:10   05/12 11:09
/usr/nr/var/new   0       0
/usr/nr/var/old   53      354256   05/06 18:05   05/12 10:49
/usr/nr/var/iplog 0       0
/usr/nr/var/dump  0       0
```

Using FM_Action

The FM_Action instrumentation shows the current configuation of actions. To execute it, use the following command line:

```
nrget 10007 . . 1 FM_Action
```

This will display the following:

```
Action Configuration
Action          system command (args)
--------------  ----------------------------
DBLoad_Proc     DBLoad ($FileOldest )
Batch_Proc      /usr/nr/bin/sap/batch1 ($DirPath $Notify1 $DirUsed)
Archive_Proc    /usr/nr/bin/sap/test1 (ARG1 $DirPath )
```

Now that you know how to use SAP instrumentation, the next step in the tutorial is learning how to use triggers.

Triggers

sapd bases all of its actions on its trigger configuration. A trigger consists of a **condition** and an **action**. The condition defines **when** to launch, the action is **what** to launch. During the course of this discussion, you'll learn how to create and customize a trigger.

As stated earlier in the *File Management Thresholds* section, there are 5 different types of conditions used to initiate a launch; three are filesystem-based, and two are time-based, as illustrated in Table 5.2.

Table 5.2: Conditions

Basis	Name	Token	Meaning
Filesystem	DirFiles	FM_DirFiles	Number of files in location
Filesystem	DirSize	FM_DirSize	Bytes consumed in location
Filesystem	DirUsed	FM_DirUsed	Percentage of disk space used
Time	IdleTime	FM_IdleTime	Relative time (elapsed time) in minutes
Time	CronTime	FM_CronTime	Absolute time (depends on day & time) in minutes

The command-line syntax for setting a condition is as follows:

```
FM_<token>   LABEL   VALUE   LOCATION   ACTION
```

Where **LABEL** is a user-defined name; **VALUE** represents either number of files, number of bytes, percentage of disk space used, or time in minutes; **LOCATION** indicates a directory; and **ACTION** specifies the name of the action to be run.

For example, you could set the following condition:

```
FM_DirFiles  OraLoad  2  /usr/nr/var/new  OracleLoad
```

This means that when the number of files in the directory /usr/nr/var/new reaches two, then launch the action OracleLoad. As you can see, the VALUE placeholder (the "2" in our example) maps to the Meaning column in the above table.

After setting a condition, you'll need to set a corresponding action. The command line is very much like that for setting a condition, except you'll more than likely be using variables as your arguments. The syntax for setting an action is as follows:

```
FM_Action   ACTION  EXEC [ARGS]
```

As an example, type in a command that will set a corresponding action to the condition created above:

```
FM_DirFiles   OracleLoad   DBLoad $DirPath $FileOldest
```

The **ACTION** is a user-defined name, and should match the ACTION variable in the Condition for the trigger to work properly. The **EXEC** is the script that is executed—in the case of our example, it is DBLoad. The **ARGS** (or arguments) are literal parameters, unless you use the reserved macros listed in Table 5.3. In the case of our example, `$DirPath` will be replaced by the /usr/nr/var/new of the Condition, and `$FileOldest` will look for the filename of the oldest file in that directory.

You have just created your first trigger. Let's go over what the trigger does:

1. **The Condition stipulates that OracleLoad should run only if the /usr/nr/var/new directory contains two files.**

2. **The Action stipulates that OracleLoad will look in the /usr/nr/var/new directory for the oldest file, and perform a DBLoad.**

For more information on DBLoad and other *sapd* Actions, please refer to page 10.

Table 5.3: Macros and Expanded Values

Macro	Expanded Value
Filesystem Macros	
$DirFiles	count of all files in trLoc
$DirSize	size in bytes of all files in trLoc
$DirUsed	disk percentage used where trLoc resides
$DirPath	directory location
$FileOldest	filename of oldest file in trLoc
$FileNewest	filename of newest file in trLoc
Token Value Macros	
$Notify1	token value for NotifyPerson
$Notify2	token value for NotifyPerson2
$DBUser	token value for DBUser
$DBPass	token value for DBPass
Date/time Macros	
$TimeStamp	standard date/time stamp *(19970531.0137)*
$CurrTime	current time string *(23:37)*
$Today+1	date of tomorrow *(1997/05/31)*
$Today	date of today
$Today-1	date of yesterday
$Today-2	date of day-before-yesterday
$Today-3	date of day-before-the-day-before-yesterday
$Week	date of day a week ago
$Week-1	date of day 2 weeks ago
$Month	date of day starting this month
$Month-1	date of day starting the previous month

How to Configure a Trigger

Configuring a trigger to the needs of your organization is a three-step process:

1. **Edit the tokens in /usr/nr/etc/sapd.conf. They are easily recognized because they all begin with FM_ (e.g., FM_DirFiles, FM_DirSize).**

2. **After you've edited sapd.conf, either perform an nrstart (if you're not currently up) or an nrhup (if you're already up—this will activate the edited values in sapd.conf).**

3. **Finally, perform an nrget on both FileMgmt (in either Brief or Full mode) and FM_Action to view your settings and ensure that everything is correct.**

How to Change the PollingDelay

The speed at which *sapd* evaluates its triggers is controlled by the PollingDelay token. The valid values range from 1 to 9. The lower the value, the faster the polling. A value of 1 means *sapd* will evaluate the triggers every 10 seconds; a value of 9 means *sapd* will evaluate the triggers every 90 seconds.

If you want *sapd* to launch actions as quickly as possible, set the PollingDelay to 1. Use this if you have many small files in the /usr/nr/var/new area and you want them loaded quickly. If you want *sapd* to slow down launching of actions, set the PollingDelay to a larger value.

Use the following command-line syntax to change the PollingDelay on the fly:

```
nrset PollingDelay <1-9>
```

To set the PollingDelay permanently, edit sapd.conf, simply type the following:

```
PollingDelay <1-9>
```

SAP Reference

This section builds on the knowledge contained in the *SAP Tutorial* section. If you have not read the tutorial section yet, please do so now.

This section covers the following information:

- Thresholds and Triggers
- *sapr*
- Scripts and skel
- Operations

Thresholds and Triggers

This subsection discusses various ways to tune the thresholds in the /usr/nr/var file locations.

Remember that *loggerd* defines the PUSH threshold with **NumberOfSwitchBytes** and **NumberOfSwitchMinutes** tokens, and *sapd* defines various PULL thresholds with its filesystem and date/time condition tokens.

Do you have enough space?

The /usr/nr/var area must be big enough to include the data in the ~/new, /old, /dump, and /iplog directories, as well as a padding space of 100 megabytes. This padding is set as a default; the needs of your system may require more or less padding space to accomplish file management and push-pull processes.

The following formula is presented as a checksum to ensure that enough file space has been allocated:

```
nr/var size  > varTrap + {newTrap | newLoad || newArch}(_____) +
iplogArch + dumpOffline(_____) + padding(100MB)
```

Do you have enough room left over for archiving?

The **DirSize** token identifies how much of a disk partition can be dedicated to logging before an archiving action (such as compression or file deletion) can occur. It is a good idea to set the **DirUsed** token to a variable low enough to ensure that enough space is left to accomplish archiving. For example, setting the **DirUsed** token to 90 percent on a one gigabyte disk partition may leave enough room (100 MB), but if you are using a 500 megabyte disk partition, you may want to set the **DirUsed** token to 65 percent.

How do you want your data batched?

If you want fast pulling of files from a directory, set **DirFiles** to 1. To slow down pulling, set the **DirFiles** to a higher number, like 10. This will effectively change how many different files get batched together. Similar effects can be had with time thresholds. Setting **IdleTime** to a lower number will speed up polling, and setting it higher will slow polling down.

sapr

Reports

The SAP queries can be run either interactively or in batch mode. "Interactive mode" is useful for the most up-to-date information and for specific queries, but they take longer to run. "Batch mode" supports the "Ready Reports" but will only include information present in the database at the time of report generation. Please refer to the *Database Reports Setup and Customization* section for a sample interactive query and the results.

Customizing Queries

The SQL source code can be found in /usr/nr/bin/sap/sql/skel. This includes the queries that initialize the database schema for NetRanger tables as well as the queries used to analyze NetRanger data. Table 5.4 lists the included queries, which serve as starting points for query building.

Table 5.4: SQL Queries

Space Dimension Queries (space.sql)	
space1.SQL	source signature summary
space2.SQL	destination signature summary
space3.SQL	connection signature summary
space4.SQL	connection signature with string data summary
Time Dimension Queries (time.sql)	
time1.SQL	timeview signature summary
time2.SQL	timeview source signature summary
time3.SQL	timeview destination signature summary
time4.SQL	timeview connection signature with string data summary
Event Dimension Queries (event.sql)	
event1.SQL	alarm level summary
event2.SQL	alarm signature summary
event3.SQL	alarm signature with string data summary
System Queries (system.sql)	
system1.SQL	age and count of records in each table
system2.SQL	signature: name-number (interactive only)
system3.SQL	organization: name-number (interactive only)

SQL Database Information

Default Schemas

The native Oracle schema supported by *sapx* and *sapr* is illustrated in the following diagram. Each box on the left identifies how event data is grouped prior to being loaded into a database. Each box on the right represents the target database tables. The schemas for these tables are defined in Tables 5.5 through 5.9.

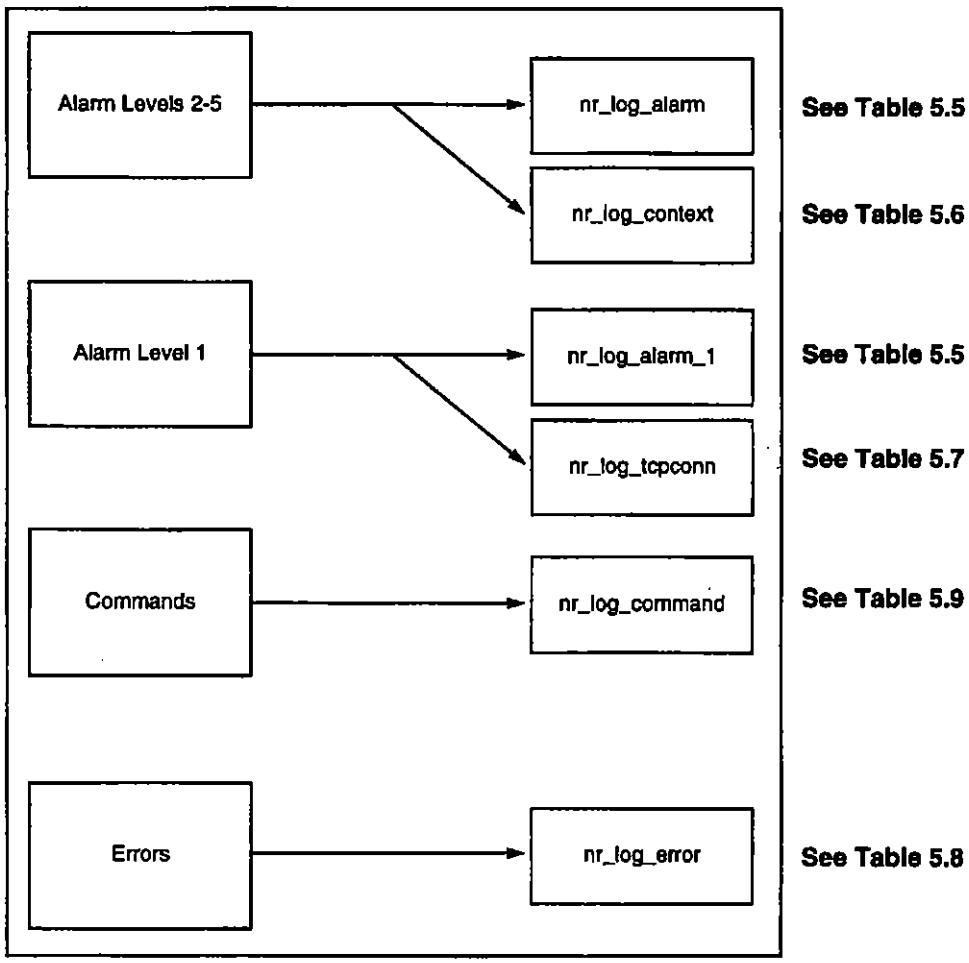

Table 5.5: nr_log_alarm and nr_log_alarm_1

Name	Type
EVENT_PROTOCOL	NUMBER(5)
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE_LOCAL	DATE
APP_ID	NUMBER(5)
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
FROM_STATE	CHAR(1)
TO_STATE	CHAR(1)
EVENT_LEVEL	NUMBER(5)
IP_SIGNATURE	NUMBER(10)
IP_SUB_SIGNATURE	NUMBER(10)
NETWORK_PROTOCOL	CHAR(3)
SRC_IP_ADDR	CHAR(32)
DST_IP_ADDR	CHAR(32)
SRC_PORT	NUMBER(5)
DST_PORT	NUMBER(5)
ROUTER_IP_ADDR	CHAR(32)
DATA_ALARM	CHAR(64)

Table 5.6: nr_log_context

Name	Type
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
DATA_MATCH	CHAR(64)
DATA_INCOM	VARCHAR2(768)
DATA_OUTGO	VARCHAR2(768)

Table 5.7: nr_log_tcpconn

Name	Type
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE_LOCAL	DATE
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
IP_SUB_SIGNATURE	NUMBER(10)
SRC_IP_ADDR	CHAR(32)
DST_IP_ADDR	CHAR(32)
SRC_PORT	NUMBER(5)
DST_PORT	NUMBER(5)
ROUTER_IP_ADDR	CHAR(32)

Table 5.8: nr_log_error

Name	Type
EVENT_PROTOCOL	NUMBER(5)
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE_LOCAL	DATE
APP_ID	NUMBER(5)
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
DATA_ERR	VARCHAR2(256)

Table 5.9: nr_log_command

Name	Type
EVENT_PROTOCOL	NUMBER(5)
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE_LOCAL	DATE
APP_ID	NUMBER(5)
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
SRC_APP_ID	NUMBER(5)
SRC_HOST_ID	NUMBER(10)
SRC_ORG_ID	NUMBER(10)
DATA_CMD	VARCHAR2(256)

Scripts and skel

The DBLOAD process relies on two types of scripts: control and conversion. Control scripts move data through the SAP system. Conversion scripts transform data into new output formats, such as cpio tape formats, printer hard copy, and relational database tables.

skel is a directory under /usr/nr that contains templates that specify alternate loading procedures for database types other than Oracle and Remedy ARS. These templates are effectively schema definitions for alternate bulk loaders.

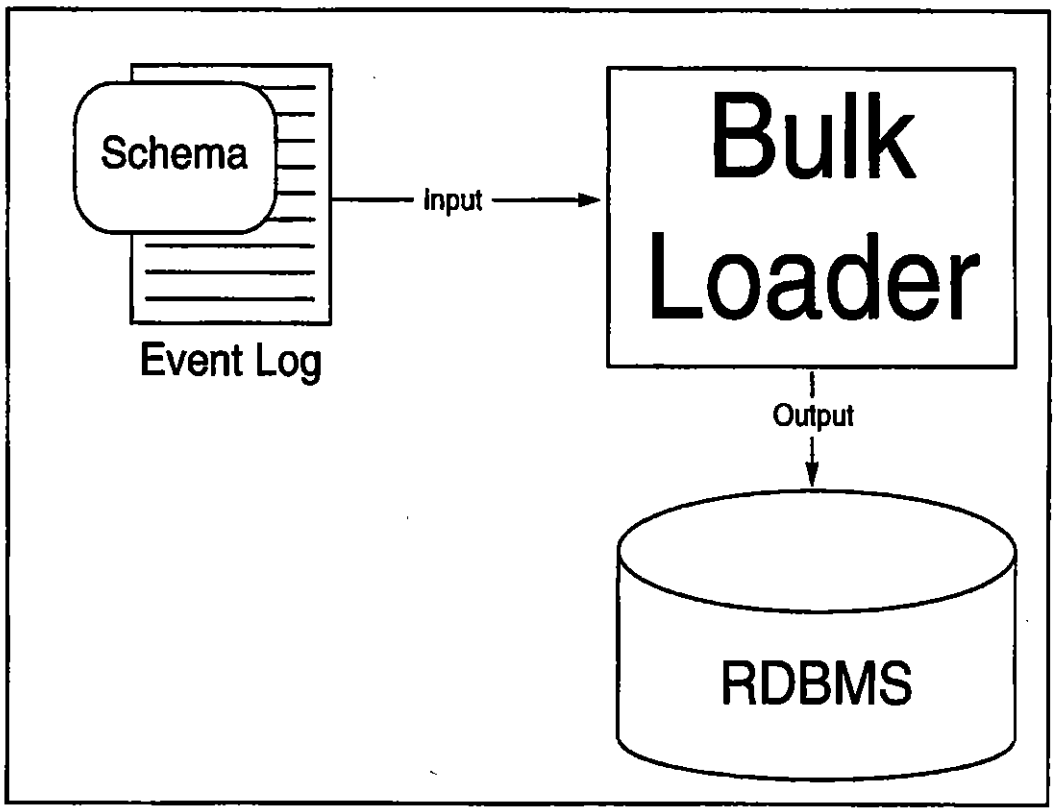

Figure 5.5: Bulk Loading Process

Alternate Loading Procedures

Defining what gets Loaded

sapx, by default, attempts to load error, command, and alarm record types. If you do not want one of these record types to be exported to your target database, modify the "for filetype in" command in the script /usr/nr/bin/sap/load_run.sh.

For example,

```
for filetype in 2, 3, 4
```

can be modified to receive only alarms by deleting the entry types for "2" (commands) and "3" (errors):

```
for filetype in 4
```

Changing the Target Database from Oracle to Remedy ARS

Replace the control script in /usr/nr/bin/sap/load_run.sh with the control script in /usr/nr/bin/sap/skel/load_run.sh.remedy.

Alternate Database Schemas and Destinations

Use the event log templates in /usr/nr/bin/sap/sql/skel as a guide for changing the schema of the database load process, or as a guide for exporting event log data to databases other than Oracle and Remedy ARS. Refer to the following for further information:

- Use the create_log*.SQL files as a template for the schema declaration.
- Use the oraldr_log*.ctl files as a template for SQL*Loader field mappings.
- Use the load_run.sh.oraldr as an example load_run.sh control script.

One of the requirements for the DBLOAD scripts is that they must return an exit status. Otherwise, *sapd* will not be able to determine when an error has occurred or perform an UNDO. Use the following settings for returning an exit status:

- If the script completed without error, return 0
- If an error was encountered during processing, return 1 or greater

NSOC Operations

Table 5.10 suggests scripts to help perform various NSOC-related tasks.

Table 5.10: Suggested scripts

Script	Task	Script Contents
Check Status		
sapdStatus	Display brief status	`nrset 10007 . . . DisplayMode Brief` `nrget 10007 . . . FileMgmt`

sapdStatusFull	Display full status	`nrset 10007 . . . DisplayMode Full`
		`nrget 10007 . . . FileMgmt`
		`nrget 10007 . . . VarSize`

sapdConfig	Show all configurations	`nrget 10007 . . 1 FileMgmt`
		`nrget 10007 . . 1 FM_Action`
		`nrget 10007 . . 1 DBConfig`
		`nrget 10007 . . 1 NotifyPerson`
		`nrget 10007 . . 1 NotifyPerson2`
		`nrget 10007 . . 1 NotifyInterval`
		`nrget 10005 . . 1 NumberOfSwitchBytes`
		`nrget 10005 . . 1 NumberOfSwitchMinutes`
		`nrget 10005 . . 1 MinContextLevel`

Make Adjustments		
sapdFast	Accelerate Polling	`nrset 10007 . . . PollingDelay 1`
sapdSlow	Decelerate polling	`nrset 10007 . . . PollingDelay 9`
sapdOK	Reset token status	`nrset 10007 . . . UserStatus OK`

sapdConfNew	Replace/ update configuration	`cp /usr/nr/etc/sapd.conf /usr/nr/etc/sapd.conf.prev`
		`cp /usr/nr/etc/sapdxxx.conf /usr/nr/etc/sapd.conf`
		`nrexec 10007 . . . ReadFileConfig`

sapdConfSave	Write active token files to sapd.conf	`nrexec 10007 . . . WriteFileConfig` `cat /usr/nr/etc/sapd.conf`
Serializing Log File		

loggerdSwitch	Switch, serialize, or execute DBLoad	`nrget 10007 . . . FileMgmt`
		`nrget 10007 . . . VarSize`
		`nrexec 10005 . . . SwitchFileLog`
		`nrget 10007 . . . VarSize`
		`sleep 60`
		`nrget 10007 . . . FileMgmt`
		`nrget 10007 . . . VarSize`

DBA Operations

Oracle database administrators are responsible for maintaining the SAP-default database. Some of the techniques described below will apply to other DBMS facilities. Once the database facilities are in place, the DBA typically will monitor the data areas, keeping operations smooth.

The DBA has two primary functions:

- Fix storage problems in the Oracle data files.
- Purge old data from database.

Fix Storage Problems

Table 5.11 illustrates recovery procedures for problems in the different Oracle data files.

Table 5.11: Recovery Procedures for Oracle Data File Problems

Area	Problem	Fix
DATA	Table storage overflow	• Extend DATA storage • Purge data
TEMP	Detailed query caused an overflow	• Extend TEMP segment • Tune TEMP segment • Filter more data from the selected DML
RBS	Purge overflows RBS	• Expand the size of your RBS • Refer to the PURGE section for alternate methods

Purging Data

Because of extended database operations, collected data will eventually outgrow the size of your Oracle disk partition. For this reason, you will need to occasionally purge the data from the nr_log_* tables.

The method you use to purge the data depends on your schema and the extent of your Oracle facilities (i.e., a large RBS will allow mass deletion, whereas a smaller RBS would require incremental and localized deletions).

A purge should not be wholesale—instead, selectively purge data from your tables. Save important alarm information and dispose only of the old or irrevelant data. Use the following guidelines as a basis for your purge strategy:

1. **First, generate some reports from the data you are ready to delete.**
2. **If you are logging Level 1 or Context data, purge these first.**
3. **Use the system1.SQL and event2.SQL queries to view the distribution of the data in your database. These queries will show the count of records in each table as well as signature-based distribution of records.**
4. **Selectively purge records from the nr_log_alarm tables.**

To remove **all** the records in a table:

1. **Drop the nr_log_alarm_1 table.**
2. **Use create_nr_log_alarm_1.sql to recreate the table.**

To remove **some** records based on time, use the following command line:

```
delete from nr_log_alarm where event_date_gmt < to_date('01/01','MM/DD')
```

To remove **some** records based on signature, use the following command line:

```
delete from nr_log_alarm where event_level < 4 and ip_signature = 8000
```

The above command lines can be combined to delete some time- and signature-based records. You can also use other fields, such as src_ip_addr (Source IP Address) to delete records from an individual machine. In other words, there is no need to use just time- and signature-based constraints in a purge, although they are the most commonly used.

NOTE

Remember to commit after the purge commands complete.

Do not use drop_nr_tables.sql for the purge because it drops **all** tables, essentially wiping your database clean.

An automated approach to data purge would be to schedule a Daily CronTime Batch trigger in *sapd* that enters sqlplus and executes report and purge DML actions.

File Management Tokens

Table 5.12: FileMgmt Tokens

Tokens	Default
Tokens for Directory Paths[a]	
LogFilePathNew	/usr/nr/var/new
LogFilePathTmp	/usr/nr/var/tmp
LogFilePathOld	/usr/nr/var/old
LogFileMask	log.
LogFilePathDump	/usr/nr/var/dump
LogFilePathIPLog	/usr/nr/var/iplog
LogFilePathVar	/usr/nr/var
Tokens for DBLOAD Launch Scripts[b]	
ControlPre	/usr/nr/bin/sap/load_pre.sh
ControlRun	/usr/nr/bin/sap/load_run.sh
ControlPost	/usr/nr/bin/sap/load_post.sh
ControlUndo	/usr/nr/bin/sap/load_undo.sh
Oracle Tokens[c]	
DBUser	username for Oracle access
DBPass	password for DBUser
Tokens for Remedy or other DBMS Default[d]	
DBUser2	username
DBPass2	password for DBUser2
DBAux1	auxiliary environment information
DBAux2	auxiliary environment information
DBAux3	auxiliary environment information

a Using these defaults is recommended, but they can be changed if necessary.

b These tokens can be changed, either permanently or on the fly to facilitate custom operations. See documentation for *configd* and Java configuration for information about token operation.

c These must be set before you can connect to Oracle. They are set to match whatever user/password was specified in the DBMS setup (create user).

d These tokens are included to provide extra flexibility for non-Oracle databases.

TROUBLESHOOTING

Overview

Sometimes things go wrong. Components often display confusing and unexpected error messages, or behave outside their normal parameters. Identification of these symptoms, their possible causes, and their possible solutions could potentially be a time-consuming and frustrating experience.

Ideally, the material in this appendix will provide a framework to help you identify and resolve routine problems. When problems are not so routine, this appendix will also help you by ruling out possibilities before you call WGC's Technical Support. This will ensure a faster and more beneficial resolution to your problem.

This Troubleshooting appendix consists of the following sections:

- Director Not Running
- Director Running
- Connectivity
- NSX
- Oracle
- Security Analysis Package

Director Not Running

Symptom	Possible Cause(s)	Possible Solution(s)
The following error message is displayed when trying to start the Director system from an HP Terminal session: `Cannot write message to Director, errno =2`	This error occurs when *smid* tries to write to a socket that does not exist. This may occur because the Director's nrdirmap application has not created its communication socket in /usr/nr/tmp because the OpenView user interface (ovw) was not started.	First ensure that the underlying NetRanger services, such as *postofficed* and *smid* are running. The easiest way to accomplish this is to execute the following from the console command line: `/usr/nr/bin/nrstatus` Run /usr/nr/bin/nrstart as user netrangr if these services need to be started. Then bring up the OpenView user interface (ovw) by executing `$OV_BIN/ovw &`. This will automatically start nrdirmap.
The following error message is displayed when trying to start the Director system from an HP Terminal session: `Cannot write message to Director, errno = 233`	This error message is generated when *smid* writes to a socket whose buffer is overflowing. This can occur when the Director's nrdirmap application is not running.	Ensure that the OpenView user interface (ovw) is running by executing `$OV_BIN/ovw &`. This will automatically start nrdirmap.
The following error message is displayed when trying to start the Director system from an HP Terminal session: `Cannot write message to Director, errno = 239`	This error message occurs when *smid* and nrdirmap do not have adequate permissions to talk to one another via sockets in /usr/nr/tmp.	Ensure that the *smid* process is owned by user netrangr, and that nrdirmap runs as suid netrangr.

One of the following error messages is generated by HP OpenView upon startup:
`848967818198: WgcSema can't access`
`'/usr/nr/etc/nrdirmap.semaphore'`
`Error: nrdirmap:main, semaphore initialization failed.`
	The nrdirmap.semaphore file in /usr/nr/etc has been deleted or has improper permissions.	Ensure that /usr/nr/etc/nrdirmap.semaphore exists and that the netrangr *user* and group accounts have read access to the file. If the file does not exist, you can create the file with an ASCII editor, such as ***vi***. Add a single character to the file and save it in /usr/nr/etc with the name nrdirmap.semaphore.
		Also ensure that /usr/nr/bin/nrdirmap runs as suid netrangr.

Symptom	Possible Cause(s)	Possible Solution(s)

`nrdirmap: fatal: libovw.so.1: can't open file: errno=2`	The LD_LIBRARY_PATH environment variable is not set properly in your user environment. Note that this may indicate that you are logged onto the Director platform as the wrong user.	Follow the instructions in Chapter 3 for setting up an HP OpenView environment for user accounts other than netrangr.
		
		If the user account is based upon either the bourne or korn shell, the following lines should exist in the user's $HOME/.profile:
		
		`if [ -d /opt/OV ] ; then`
		`. /opt/OV/bin/ov.envvars.sh`
		
		`PATH=$OV_BIN:$PATH`
		`export PATH`
		
		`LD_LIBRARY_PATH=$OV_LIB:$LD_LIBRARY_PATH`
		`export LD_LIBRARY_PATH`
		`fi`
		
		If the user must use a shell other than ksh, then the lines above must be translated into the appropriate scripting language and placed in the appropriate startup file.

Director Running

Symptom	Possible Cause(s)	Possible Solution(s)
The Director's security map contains an NSX icon, but fails to show any events for that NSX.	The Director's **Severity Status** attributes are set higher than the level of alarms being generated by the NSX.	From the Director interface, highlight the icon for the NSX system and then either press **Ctrl+O** or select **Edit→Describe→Modify** from the menu bar. Then select **NetRanger/Director** and press **View/Modify**. Ensure that the *Minimum Marginal* and *Minimum Critical* status thresholds are low enough to register events from the NSX in question.

	The level of alarms generated by the NSX system fall below the **routing threshold** set in the /usr/nr/etc/destinations file.	If the Director Severity Status thresholds are set properly, ensure that the routing threshold in the NSX's /usr/nr/etc/ destinations file is set low enough to route information to the Director. In the following example, the destinations file has the Director's *smid* event level set to "4", which means that only alarm events of level 4 and 5 will be forwarded onto the Director platform running on director1.wheelgroup:
		`1 datastore.wheelgroup loggerd 1 EVENTS,ERRORS,COMMANDS`
		`2 director1.wheelgroup smid 4 EVENTS`
		The Director should begin displaying alarms properly if you reset datastore's alarm threshold from "4" to "2".

Symptom	Possible Cause(s)	Possible Solution(s)
An NSX's alarms are properly displayed on the Director security map, but information on those alarms does not appear in the **Show Current Events** window **and** the event log file in the Director's /usr/nr/var directory does not contain any records from that NSX.	The Director *loggerd* daemon is not listed as a destination in either the NSX's or the Director's configuration files.	You can either create an entry in the NSX's /usr/nr/etc/destinations file for the Director's *loggerd* service, or you can create a **DupDestination** entry in the Director's /usr/nr/etc/smid.conf file to redirect event data to *loggerd* from *smid*. The latter would look like this: `DupDestination <HostId>.<OrgId> loggerd 1 EVENTS,ERRORS,COMMANDS` Where `<HostId>.<OrgId>` are the Director's values (e.g., director1.wheelgroup). **Note:** The first option requires that the NSX's destination file contain two entries for the Director system: one for *smid* and another for *loggerd*. This doubles the amount of network traffic that must be sent to the Director. Although the second option will create some overhead for *smid*, it is the preferred configuration, because it only requires one copy of event data to be sent over the communication pathway.
Information is properly displayed in the Director's **Show Current Events** window, but the mouse pointer turns into an hourglass and never changes back.	The current events utility continues to pull information from the Director log files as long as the window is up.	Press the **Stop** button to terminate the filtering application. You can then use the scrollbars and menu options to look at the data. Press the **Close** button to exit this window.

Connectivity

Symptom	Possible Cause(s)	Possible Solution(s)
Unable to access an NSX sensor or any of the NetRanger services running on the system.	Can you ping the NSX Sensor? **Yes**—This means that the NSX Sensor is powered up but that something is wrong with the NSX services. **No**—The BorderGuard is shunning the Director. This can occur when: • an attack is run from the Director system (as part of a system test or demonstration) that causes the NSX to shun the Director system. • the NSX does NOT contain a **`NeverShunIPAddress`** entry for the Director system.	Telnet onto the box as user netrangr and run **`/usr/nr/bin/nrstop`**. Try to deduce why services were disrupted by examining whatever error files exist in /usr/nr/var. If errors are found, refer to the appropriate Troubleshooting section, or contact WheelGroup Technical Support. Once the cause has been identified, restart the NSX services by executing **`/usr/nr/bin/nrstart`**. Gain access to the NSX system to either issue an unshun command via local execution of nrset, or restart *managed*, which will clear all current shuns on the BorderGuard. In either case you need to gain access to the NSX sensor via one of the following means: • connect to the NSX sensor via its modem • connect to the NSX via its COM1 serial port • connect to the NSX system's BorderGuard via the BorderGuard's console Although the first option provides remote access to the NSX system, the second and third options require physical access to the NSX. Option b also requires the use of a notebook computer or a terminal device.

NSX

Symptom	Possible Cause(s)	Possible Solution(s)
Unable to start or stop the NetRanger daemon processes when running **nrstart** or **nrstop**.	You are trying to run these utilities from an account that does not have access rights to the NSX daemons.	Ensure you are logged onto the NSX or Director systems under the same user account that was used to start its daemon services. (The default is user netrangr.)
/usr/nr/var/errors.smid contains multiple instances of the following error message: `socket received data from unknown host.`	The NSX system's *sensord* is receiving **copy_to** messages from a BorderGuard or Passport security device that has not been properly defined in /usr/nr/etc/sensord.conf.	Properly define a **RecordOfDataSource** for the BorderGuard or Passport device that is trying to send **copy_to** messages to the NSX system. Refer to Chapter 3 for information on properly defining the **<IP address>** and **<IP mask>** values for this sensord.conf token.

Oracle

Symptom	Possible Cause(s)	Possible Solution(s)
Cannot determine if Oracle is installed.		Check local and mounted file systems using commands `df`, `mount`, and `find`. Look for `oracle` and `product/7*`.

Cannot determine if Oracle is running.		Use the following command:
		`ps -ef | grep ora`
		After entering the command, the following should appear:
		`oracle 360 1 0 Jun 14 ? 0:01 ora_pmon_NRDB`
		`oracle 362 1 0 Jun 14 ? 4:36 ora_dbwr_NRDB`
		`oracle 364 1 0 Jun 14 ? 3:13 ora_lgwr_NRDB`
		`oracle 366 1 0 Jun 14 ? 0:40 ora_smon_NRDB`
		This means that there is an Oracle instance of SID=NRDB running since June 14. The user account oracle is the owner of these processes.

Oracle Installer (orainst) was unable to find any products to install.	**`start.sh`** was not run prior to starting orainst.	Run start.sh to prepare your environment for the orainst program. Look for the following file: `/cdrom/cdrom0/orainst/start.sh`

`sqlplus: not found`
or
`sqlldr: not found`
	The Oracle bin directory is not present or specified properly in your `$PATH`.	Set `$PATH` to include `$ORACLE_HOME/bin`.

The following error message is displayed when you try to run sqlplus: `~~~/oracle/product/7.3.2/bin/sqlplus: cannot open`	The shell finds sqlplus, but it cannot be executed. This can occur when your `$PATH` includes references to the wrong versions of the Oracle binaries. For example, you have mounted the wrong Oracle directories from a file server. Therefore, you are trying to execute HPUX binaries on a SPARC system.	Ensure that the `$ORACLE_HOME` directory contains the proper binaries for the platform you are running. Refer to the section entitled *Remote Setup: heterogeneous/homogeneous* in Appendix D.

Symptom	Possible Cause(s)	Possible Solution(s)

sqlplus, sqlldr, or *sapx* fail with the following SID error message:
`ERROR: ORA-01034: ORACLE not available`
`ORA-07200: slsid: oracle_sid not set.`
	The `ORACLE_SID` environment variable, which identifies which database **instance** to use, was not set properly prior to starting sqlplus.	Set `ORACLE_SID=NRBIG`.

sqlplus, sqlldr, or *sapx* fails with the following libc error message:
`libc.so.xxx: can't do something`
	$ORACLE_HOME/lib is not part of the `LD_LIBRARY_PATH` environment variable.	Add `ORACLE_HOME/lib` to the `LD_LIBRARY_PATH` environment variable. If you are running either the bourne or korn shell ensure that your $HOME/.profile contains the following entries:
		`LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib`
		`export LD_LIBRARAY_PATH`

sqlplus, sqlldr, or *sapx* fail with the following TNS error message: `ERROR: ORA-12154: TNS: could not resolve service name`	Oracle cannot understand the name specified in your connect string.	Ensure that the Oracle file tnsnames.ora resides in its proper location (usually $ORACLE_HOME/admin/network) and that it is properly formatted. Then use the tnsping utility to test sqlnet connectivity to your remote database.

sqlplus, sqlldr, or *sapx* return a TNS or USER/PASSWORD error message.	Improper connect string.	Correct the syntax on the connect string. Connecting from the shell prompt, do one of the following:
		If specifying the password on the command line, type:
		`sqlplus <user>/<password>@<host>`
		Otherwise, type:
		`sqlplus <user>@<host>`
		and sqlplus will prompt for a password.

Security Analysis Package

Symptom	Possible Cause(s)	Possible Solution(s)
Instrumentation shows proper configuration, but no actions are performed.	If there are no FM_Action items in sapd.conf, then nrConfig did not properly copy files into sapd.conf due to upgrade from 1.2.x to 1.3.x, or because nrConfig was not run at all.	Manually copy the sapd.conf.nsx or sapd.conf.director from /usr/nr/etc/wgc/templates into /usr/nr/etc/, replacing sapd.conf. **Note:** The file /usr/nr/etc/wgc/templates/sapd.conf contains descriptions of the tokens used by *sapd*. This file does not contain any "real" token values. It is intended as a reference for setting triggers in /usr/nr/etc/sapd.conf
There are extraneous files in the /usr/nr/var directory structure, and there exists a /usr/nr/var/old directory.	You have upgraded from SAP 1.2.x to 1.3.x, which does away with the /usr/nr/var/old directory. The upgrade has also left behind many files in /usr/nr/var.	Clean the extraneous files, delete the /usr/nr/var/old directory, and archive the /usr/nr/var/dump directory.
SQL queries do not display data.	You did not enter % as a wildcard.	Use % as a wildcard in your SQL queries.
From the SQLPLUS prompt, entering @event1, @space1, @time1, or @system1 does not return proper data.	You are using the wrong command line parameter.	SAP 1.3.x requires that you enter either @event, @space, @time, or @system. You will be queried for the desired drill-down level (e.g., 1, 2, 3).
Queries do not display new signatures.	You have upgraded from SAP 1.2.x to SAP 1.3.x without updating the nr_sigs and nr_orgs tables.	Update the nr_sigs table with /usr/nr/etc/signatures and the nr_orgs table with /usr/nr/etc/organizations. Use the code at the end of `nrdb_master_create` for this purpose.

FILTERS AND BRIDGE CONFIGURATION B

This section lists and explains the settings for all of the filter (fil), command (cmd), and data privacy facility (dpf) files required to deploy the BorderGuard or Passport as part of an NSX system. Please note that many of the settings described in this section have been automatically set for you by the nrconfig script shipped with NetRanger under the /usr/nr/bin directory. This information is presented here in the event you need to manually edit these files. Refer to the appropriate NSG BorderGuard Reference Manual for additional information about these files.

Standard Files

The standard files that must be configured and installed on a BorderGuard device are as follows:

- startup
- filters.cmd
- sleeves.cmd
- first.fil
- incom(x).fil
- last.fil
- shun.fil
- sleeves.fil
- params.dpf
- pubkeys.dpf
- sleeves.dpf

These files are discussed on the following pages.

startup—This is the boot configuration file that is read each time the BorderGuard powers up or resets. This file performs these tasks:

- Names the device.
- Compiles and applies filters.
- Configures interfaces.
- Sets up route tables.
- Executes the files required to establish encrypted sleeves.
- Establishes a nameserver.
- Starts daemons such as *telnetd*, *gated*, and *snmpd*.

filters.cmd—This file compiles all the filters that enforce your security policies. It is called in the startup file via the following command:

```
@filters.cmd
```

This file contains a single line for each filter used on the BorderGuard. Most *filters.cmd* files contain at least the following three entries:

```
ns> compile_filter first.fil
ns> compile_filter incom1.fil
ns> compile_filter shun.fil
```

Filter Templates

These files enforce security policies as well as establish filtering on the BorderGuard/ Passport. Refer to the *NetSentry User Guide* for information on how to create and apply filters. When you are editing these filters, you may find it helpful to refer back to *Chapter 3*, which you used when you defined your security policy.

first.fil—This file identifies which network traffic should be copied to the NSX sensor. First, change the IP address of the NSX that follows the `copy_to` commands if it has not already been changed. Then change any string-matching requirements defined in sensord.conf.

For string matching to be effective, all packets for a given service (e.g. Telnet at port 23) must be copied to port 35398. By default, FTP (21), Telnet (23), mail (25), DNS (53), RPC (111), and WWW (80) packets are all copied to the tcp_dp and tcp_sp command lines in *first.fil*. To set up string matching on additional services such as FTP data (20), add the ports to *first.fil*:

```
# Copy certain TCP traffic
ip_protocol in (6)
tcp_dp in (21,23,25,53,80,111,20)
copy_to IP_ADDRESS_NSX 35398 break;
```

NOTE

The port number following the NSX is 35398 instead of 35399 as it appears in the other filter files. **This is the UDP port for intrusion detection. Do NOT change it!**

Incom(X).fil—This file is the core of the NetRanger security system and it establishes the security policy defined in Chapter 3. Apply this filter to each network interface to establish a security policy. The general naming convention is `incom1.fil` for interface 1, `incom2.fil` for interface 2, and so on.

NOTE

The default templates assign `incom1` to the interface connected to your untrusted networks. Notice that the number following the `copy_to` command in this file is 35399 instead of 35398, as indicated in the previous filter file. **This is the security policy monitoring port. Do NOT change it!**

If nrconfig has not been run, the first information that should be included is the IP address of the NSX after every `copy_to` command.

```
copy_to IP_ADDRESS_NSX 35399 break;
```

Next, make changes to the *Incom(X)_ip_spoof_fail* filter. **This filter prevents and alarms on any connections from outside your trusted network that have a source address from your internal network.** When applied to your trusted network interface, it assures that no outgoing traffic is spoofing an external address. Collect all of your internal class C and class B addresses and create an entry for each one. You should also create an entry for the individual external IP address. In the case of a site with two internal protected networks with class C addresses of 199.99.99.0 and 199.99.100.0 and a router with an external IP address of 199.99.101.1, the filter attached to the untrusted network would look like the following:

```
filter INCOM1_IP_SPOOF_FAIL
ip_sa mask 0xFFFFFF00 in (199.99.99.*)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
ip_sa mask 0xFFFFFF00 in (199.99.100.*)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
ip_sa mask 0xFFFFFFFF in (199.99.101.1)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
end
```

For the filter attached to the trusted network, make the following changes:

```
filter INCOM1_IP_SPOOF_FAIL
not ip_sa mask 0xFFFFFF00 in (199.99.99.*)
not ip_sa mask 0xFFFFFF00 in (199.99.100.*)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
end
```

The next change you need to make is based on your security policy. Every allowed service should have a corresponding filter (for example, you should have a filter similar to `incom2_smtp_fail` for e-mail). You need to edit each of these filters in accordance with your policy, which typically is to allow or block network access. For each allowed service in the security policy, edit the current filter with its name. There are two basic cases for allowable services:

- **The first and simplest case is when you want to allow all services of this type through.** This is most often the case for the interface attached to a trusted network, because users want unrestricted access to e-mail and WWW services. For example, to allow all outgoing mail traffic, simply comment out the body of the filter *incom(X)_smtp_fail* as follows:

```
filter INCOM2_SMTP_FAIL
#tcp_dp in (25)
#copy_to IP_ADDRESS_NSX 35399
#icmp_unreachable fail;
end
```

  This allows all outgoing connections to port 25.

- **The second case for allowing a given service involves restricting that service to a specific destination or source IP address.** This type of restriction is usually applied to the interface attached to an untrusted network. In this case, the filter should be edited as follows:

```
filter INCOM1_ICMP_FAIL
ip_protocol in (1)
#This is where you should enter the typefields of the ICMP
#services you want to block. In this example, we are blocking
#"echo requests" type field 8. To block "timestamp requests"
#you would add 13. t1_byte 0 in (8,13)
t1_byte 0 in (8)
not ip_sa in (198.98.98.*)
not ip_da in (IP_ADDRESS1, IP_ADDRESS2, IP_ADDRESS3, ETC.)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
end
```

The above example allows all ICMP echo requests that are from the network 198.98.98.0 or that are destined for IP_ADDRESS1, 2, or 3. All of the filters should comply with your security policy requirements. Please refer to the *NetSentry User's Guide* for more information on this topic.

llast.fil—Although this filter is not required, it provides a convenient way to permanently block a particular site. This type of filter might look like this:

```
filter LAST_BLOCK_FAIL
ip_sa in (BLOCKED_IP_NETWORK)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
end
```

shun.fil—NSX system uses this file to **dynamically block** outgoing as well as incoming IP addresses. It is not applied to a BorderGuard/Passport until the NSX generates an alarm for an attack signature that is configured to shun.

NOTE

The name of this file should NOT be changed. The only thing that needs to be edited in this file is the IP address of the NSX that follows the `copy_to` command. If you have run *nrconfig*, this should already be changed.

NetRanger and Bridging

Below is a sample startup file for a BorderGuard 2000 routing on four interfaces. First, the name is set, then the *filters.cmd* script is run (this compiles the NetSentry filters). The first_filter is then applied on the IP first filter point. The four interfaces are then started, followed by starting the Telnet daemon on the BorderGuard. Each of the interfaces has certain filters applied when it is configured. These filters are on the incoming/outgoing filter points for each interface. Each of the Ethernet ports is configured at startup by the file.

```
system set name border
@filters.cmd
netsentry apply ip first_filter on first
ip start if en01 10.1.1.1 netmask 0xffffff00 incoming=incom1_filter \
outgoing=out1_filter
ip start if en02 10.1.2.1 netmask 0xffffff00 incoming=incom2_filter
ip start if en03 10.1.6.1 netmask 0xffffff00 outgoing=out3_filter
ip start if en04 10.1.4.1 netmask 0xffffff00 incoming=out4_filter
ip start telnetd
```

If the same BorderGuard 2000 were deployed in bridge mode, the important difference is that it has only 1 IP address on 1 IP network. Again, the name is set first, followed by the script to compile NetSentry filters. Here, the first_filter is applied on the MACADDR first filter point. That is the filter point for bridging. NetSentry filters are applied on each of the interface's corresponding incoming/outgoing filter points by the `set` commands as shown in the example below.

```
system set name border
@filters.cmd
netsentry apply macaddr FIRST_FILTER on first
ip start if br01 10.1.1.1 netmask 0xffffff00
bridge set port en01 incoming=INCOM1_FILTER outgoing=OUT1_FILTER
bridge set port en02 incoming=INCOM2_FILTER
bridge set port en03 outgoing=OUT3_FILTER
bridge set port en04 incoming=OUT4_FILTER
ip start telnetd
```

Unlike configuring interfaces for IP routing, the bridge configuration is automatically stored on the BorderGuard file system as configuration commands are issued. Two files are created with the names BRIDGE.DAT and BRIDGE.HDR. The command to start an interface to bridge is

```
bridge set port <interface> state on
```

The BorderGuard will immediately configure the interface by turning the port off, then on, and finally saving the configuration to disk. Here is an example:

```
BG2000 Bridge> set port en01 state on
bridge: syslog contains unread text
bridge: do 'bridge show syslog' to view
BG2000 Bridge>
%Ethernet-8-2505, 18-Feb-1997 22:10:38.966
Ethernet port en01 is OFF.

%Ethernet-8-2504, 18-Feb-1997 22:10:38.966
Ethernet port en01 is ON.

%Bridge-8-322, 18-Feb-1997 22:10:39.260
Save state complete

BG2000 Bridge> sys dir
File name            Size      Date      Time
.                   <dir>  07-Feb-1997  12:08
README              29300  15-Feb-1997  15:30
PROFILE                21  09-Feb-1997  12:51
STARTUP                62  09-Feb-1997  12:58
SHUN.FIL               19  10-Feb-1997  22:34
FIREWALL.DEF        19224  15-Feb-1997  15:30
FIREWALL.SET         7180  15-Feb-1997  15:30
BRIDGE.DAT            848  18-Feb-1997  22:10
BRIDGE.HDR             44  18-Feb-1997  22:10

9/256 files, 57100/86660 bytes used/free
42608 recoverable bytes
BG2000 Bridge>
```

Once this command is issued for an interface, it does not need to be issued again—hence the exclusion from the startup file. However, if the command is placed in the startup file, it does not cause any problems.

NOTE

Do not assign an IP address to an interface if you want it to bridge IP packets. Routing takes precedence over bridging. If a port, like en01, is configured to do bridging and it has an IP address assigned to it, the IP packets are routed and other protocols (IPX/SPX, AppleTalk, etc.) are bridged.

Filters

In bridging mode, the following should occur for NetRanger filter files:

1. **copy_to should be replaced by log_to.**
2. **UDP port 35398 should be replaced by 35400.**
3. **UDP port 35399 should be replaced by 35401.**

Here is an example of a part of the filter netranger:

```
filter NETRANGER

        ip_protocol in (1)
log_to 10.1.100.1 35400 break;

ip_protocol in (6)
tl_byte 13 mask 0x07 in (1,2,4)
log_to 10.1.100.1 35400 break;

tcp_dp in (21,23,25,53,80,111,512,513,514)
log_to 10.1.100.1 35400 break;

tcp_sp in (21,23,25,53,80,111,512,513,514)
log_to 10.1.100.1 35400 break;
...
...
...
```

Here is an example of a security policy filter:

```
filter INCOM1_IP_SOURCE_ROUTE_FAIL
ip_option_present 0x83
log_to 10.1.100.1 35401
icmp_unreachable fail;
ip_option_present 0x89
log_to 10.1.100.1 35401
icmp_unreachable fail;
end
```

DPF Files

The BorderGuard 1000 and 2000 systems are equipped with a powerful encryption capability that can establish a Virtual Private Network (VPN) between geographically remote sites over public networks.

At a minimum, WheelGroup recommends that you use encrypted sleeves to encrypt all traffic between remote networks. It is not necessary to use encrypted links between internal NetRanger components whose traffic is only crossing internal protected private networks.

A good method to determine the necessity for encrypted sleeves is to trace the path of alarms and traffic traveling between two NetRanger components and consider the security of the networks that are crossed. If you feel comfortable with those networks, it is probably not necessary to use encryption and incur the performance reduction on traffic traveling within the sleeve.

Please note that the Passport system currently does not provide any type of VPN Facilities.

Refer to the NSG manual, *Data Privacy Facility User's Guide*, before attempting to make changes to any of the files in this section.

pubkeys.dpf—This file contains the public keys for the BorderGuard. This file should have an entry for the local system and each remote BorderGuard for which an encrypted sleeve is maintained.

NOTE

sleeves.dpf requires long keys. Be sure that you generate long public keys for both ends of the encrypted sleeve.

sleeves.dpf—This file contains sleeve definitions for the BorderGuard's DPF. During initialization, it is loaded as a set of sleeve definition commands and placed into the DPF interface's working storage. The default sleeve settings rarely need editing. The only entries that should be edited are sleeve definitions. The format for these entries is

```
sleeves_<orgId1>_<orgId2>_<sleeve_num> group1 <IP> group2 <IP>
```

where `orgId` corresponds to the organization ID established in the /usr/nr/etc/organizations file discussed in *Appendix E*. `orgId1` is the organization at one end of the sleeve and `orgId2` is the organization at the other end of the sleeve; `orgId1` may equal `orgId2`. The `sleeve_num` corresponds to the sleeve number at a particular site. The IP addresses assigned for group1 and group2 correspond to the router/bridge interface designated at the BorderGuard at each end of the sleeve as the DPF interface.

NOTE

Sleeve definitions should be identical at both the local and remote site.

For example, consider site #1 with two BorderGuards, which have the DPF IP addresses 199.99.99.1 and 199.99.100.1, and site #2 with one BorderGuard with a router DPF IP address 198.98.98.1.

The following sleeve definitions might apply:

```
sleeves_100_101_1 group1 199.99.99.1 group2 198.98.98.1
sleeves_100_100_1 group1 199.99.99.1 group2 199.99.100.1
```

This would establish two sleeves. The first is between one of the BorderGuards at site #1 and the BorderGuard at site #2. The second is between the two BorderGuards at site #1.

NOTE

Be sure to associate the correct IP address with the correct group (`group1` or `group2`).

sleeves.fil—This file contains two filters for every DPF sleeve connection defined in *sleeves.dpf*. The first filter controls outbound traffic and the second filter checks inbound traffic.

NOTE

The **IP address of the NSX** (after the *copy_to* statements) and the **sleeve name** must match the name designated in *sleeves.dpf*. For example, if site #1 has a class C address 199.99.99.0 and site #2 has a class C address 198.98.98.0, and the sleeve name is s_100_101_1, the filter at site #1 would look something like the following:

```
filter DPF_100_101_1
ip_sa in (199.99.99.*)
set_sleeve s_100_101_1;
end;
filter DPF_FAIL_100_101_1
ip_da in (199.99.99.*)
not sleeve s_100_101_1
icmp_unreachable
copy_to IP_ADDRESS_NSX fail;
end;
```

A corresponding filter would be created for the remote site and would replace 199.99.99.* with 198.98.98.* but the sleeve names would remain the same.

sleeves.cmd—This file compiles and applies all the filters required to establish encrypted sleeves between a local BorderGuard and one or more remote sites. The filters are established in the *sleeves.fil* file and should be applied so that all traffic traveling between two sites is encrypted. For example, if site #1 had a class C address of 199.99.99.0 and site #2 had a class C address of 198.98.98.0, then the apply statements would be similar to the following:

```
ip apply DPF_100_101_1 on 199.99.99.* to 198.98.98.*
ip apply DPF_FAIL_100_101_1 on 199.99.99.* to 198.98.98.*
```

In this example, `DPF_100_101_1` and `DPF_FAIL_100_101_1` correspond to the filter names established in the file *sleeves.fil*.

Bridging and DPF

DPF sleeves can be used when the BorderGuard is running in bridge mode. DPF sleeves should be defined and established normally. However, packets should be inserted into the sleeve using an IP filter. This is required because the set_sleeve action is not supported in filters on bridging filter points. To allow IP packets to pass through an IP filter on a BorderGuard running in bridge mode, follow these steps:

1. **Establish a normal DPF sleeve, correctly configuring** pubkeys.dpf, params.dpf, **and** sleeves.dpf.

2. **Assign the IP address assigned to port br01 as the DPF address of the BorderGuard.**

3. **Define a filter to place packets into the DPF sleeve (an example is shown below).**

   All the packets coming from the 10.1.6.x network and going to the host 10.1.9.10 will be placed into the DPF sleeve qa_test.

```
filter DPF
        ip_sa mask 0xffffff00 in (10.1.6.0)
        ip_da in (10.1.9.10)
                set_sleeve qa_test;
end
```

4. **Apply the filter on the IP first filter point. (The filter point** local_in **would also work.) An example (which would be added to the startup file) follows:**

```
ns apply ip DPF on first
```

5. **Verify that none of the bridge security policy filters fail any packets destined for the sleeve.**

6. **Configure the NSX (or Director) to route packets to the other destination through the IP address assigned to port br01. An example follows:**

```
route add 10.1.9.10 <port br01 IP address> 1
```

This technique works because the NSX sends IP packets to the Director that are encapsulated in Ethernet packets with a destination Ethernet address corresponding to br01 (because it is the router). The IP packet will first go through all of the bridge filter points and then through the IP filter points because it is going to br01. The filter defined in Step 3 then takes the IP packets and puts them into the sleeve for transmission to the other side. If an IP packet that should NOT go into the sleeve is sent to br01, br01 routes that packet according to its own routing table.

When the packet comes out of the sleeve on the other end, the routing table is consulted and determines that the IP packet should be transmitted through the br01 interface, which in turn causes it to be bridged out into the network to its final destination.

NetRanger Support for Passport Bridging

NetRanger Version 1.3.1 supports the Passport in bridging mode. Two additional UDP ports have been added to nr.sensord to process the specialized log_to packets from the Passport. The table below lists the UDP ports for nr.sensord.

Socket Name	Default Port #	Supports
NCI1	35398	Intrusion Detection
NCS1	35399	Security Policy Monitoring
NLI1	35400	Intrusion Detection
NLS1	35401	Security Policy Monitoring
NLI2	35402	Intrusion Detection
NLS2	35403	Security Policy Monitoring

Socket Name Key

N—Network Systems

C—copy_to packets

L—log_to packets

I—Intrusion Detection

S—Security Policy Monitoring

#—Version Number

For example, `NLI2` indicates that support for Network Systems log_to packets for intrusion detection (version 2) is enabled.

When you enable intrusion detection support for a UDP port, packets sent to that UDP port will be used for intrusion detection and logging. A filter should be placed on the NetSentry first filter (for IP and Bridging) point to copy_to/log_to packets to the NSX for analysis.

When security policy monitoring support is enabled for a UDP port, packets sent to that UDP port will be analyzed to determine the name of the NetSentry filter that performed the copy_to/log_to. In the following example, the filter will fail certain packets and send them to the NSX:

```
filter INCOM1_FINGER_FAIL
   tcp_connect_request
   tcp_dp in (79)
       log_to 10.1.1.1 35401
       fail;
end
```

In this case, *sensord* will obtain the filter name `INCOM1_FINGER_FAIL` from the log_to packet, attempt to find a matching RecordOfFilterName entry to get the filter name SUB-SIGID, look up the corresponding alarm levels set by SigOfFilterName, and then transmit an alarm as needed. The IP packet is all but ignored except to obtain source/destination addresses/ports.

The entries in /usr/nr/etc/sensord.conf for the above security policy filter would look like the following:

```
RecordOfFilterName  1011   INCOM1_FINGER_FAIL
SigOfFilterName   1011      0   3   3   3   3
```

In prior releases of NetRanger, *sensord* only supported the first two sets of UDP sockets. The NLI2 and NLS2 sockets have been added to support the log_to packet format generated by the Passport. This format is different than the one supported by the NLI1 and NLS1 sockets for BorderGuard log_to packets.

NOTE

Extra caution must be used when creating filters to send log_to packets to the NLI2 and NLS2 sockets when in bridge mode.

Caution is necessary because the log_to header does NOT specify the type of MAC header. A Passport with Ethernet, Token Ring, and FDDI interfaces that sends log_to packets to the NSX will include differing MAC headers. The NSX cannot easily determine which MAC type is included in the log_to packet.

To deal with this problem, *sensord* looks at the NetSentry name in the log_to header to determine the type of MAC header.

The NLI2 socket REQUIRES the modification of the traditional NetRanger first filter. The filter is now divided into three sections, each of which is used to process different types of MAC packets. This ensures that the log_to header will contain one of the following three filter names for packets received at the NLI2 UDP socket:

NETRANGER_BR_ETHER	Ethernet support
NETRANGER_BR_TOKENRING	Token Ring support (not tested)
NETRANGER_BR_FDDI	FDDI support (not tested)

Following is a sample breakdown of the new filters to be used on the Passport in bridge mode:

```
filter FIRST_BR_FILTER      <-- Applied at the bridge first filter point
   filter NETRANGER_BR;
   filter SHUN;
end
filter NETRANGER_BR
   filter NETRANGER_BR_ETHER;
   filter NETRANGER_BR_TOKENRING;
   filter NETRANGER_BR_FDDI;
end
filter NETRANGER_BR_ETHER
  not mac_media_type in (1)    <-- Return from filter if not Ethernet
      break;
   ip_protocol in (1)            <-- Traditional filter
      log_to 10.1.100.1 35402 break;
   ...
   ...
   ...
end
```

The NLS2 socket requires that all security policy filters have one of the following strings prepended to the filter name to ensure that the type of packets sent to the NSX via log_to have matching MAC headers. If one of these strings listed in the table below is not prepended to the filter name, *sensord* will ignore the log_to packet.

ETHER	Ethernet support
TOKENRING	Token Ring support (not tested)
FDDI	FDDI support (not tested)

NOTE

Filter names should not exceed 31 characters in length. In some cases, prepending one of the MAC formats listed above to certain default NetRanger policy filters will result in a filter name that exceeds the 31-character limit. In those cases, truncation will be necessary.

In the following example, the security filter has been rewritten for the NLS2 socket. In this example, the filter is applied on the incoming filter point of an Ethernet interface. This ensures that all log_to packets from this filter include Ethernet MAC headers. If this filter were at another filter point (first, last, apply_table, etc.), then it would have to be broken up in triplicate like the first filter shown on the previous page.

```
filter ETHER_INCOM1_FINGER_FAIL
   tcp_connect_request
   tcp_dp in (79)
       log_to 10.1.1.1 35403
       fail;
end
```